ACL2010

Uppsala, Sweden
July 11–16, 2010

The 48th Annual Meeting of the Association for Computational Linguistics

Conference Proceedings

UPPSALA
UNIVERSITET

ACL 2010

48th Annual Meeting of the Association for Computational Linguistics

Proceedings of the Conference

11-16 July 2010
Uppsala University
Uppsala, Sweden

Production and Manufacturing by
Taberg Media Group AB
Box 94, 562 02 Taberg
Sweden

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-932432-66-4 / 1-932432-66-3 (Volume 1)
ISBN 978-1-932432-67-1 / 1-932432-67-1 (Volume 2)

Preface: From the General Chair

Welcome back to Europe at ACL 2010! After three years, the ACL crowd is meeting again in Europe, this time at the very north, to escape from the Central European heat it experienced in 2007.

This year, some significant changes can be found under the hood. The call for papers was formulated much more broadly than usual, and this idea brought up by the ACL membership and the Exec and then developed in detail by this year's program chairs, Sandra Carberry and Stephen Clark, really caught on - the number of submissions has been the highest of all times, forcing us to put some activities, such as the SRW, as the fifth track on Tuesday morning. The number of reviewers is hard to compute exactly - but a glimpse into their lists in this year's and previous years' proceedings reveals that we almost certainly set a new record here, too (thank you all!). Also, the proceedings have switched to electronic-only for all events, and adaptation of the START conference automation software has begun towards a fully automated workflow from submission to the production of the final proceedings in pdf format. It has been made possible thanks to Philipp Koehn's and Jing-Shin Chang's willingness to serve as Publication Chairs two years in a row in order to ensure a smooth transition from the semi-manual process employed in the past. However, there was one thing that overshadowed it all: the enthusiastic, meticulously precise and absolutely professional yet in every situation very polite approach of the local arrangements committee headed by Joakim Nivre. His efforts have made my job, as the General Chair, a piece of cake, limited essentially to watching the tons of emails exchanged between the local and other committees and to answering emails like "why wasn't I asked to be an invited speaker?" (obviously, from people no one would consider for this honor anyway).

Joakim has been helped by Beáta Megyesi, Rolf Carlson, Mats Dahllöf, Marco Kuhlmann, Mattias Nilsson, Markus Saers, Anna Sågvall Hein, Per Starbäck, Oscar Täckström, Jörg Tiedemann, Reut Tsarfaty and by the Akademikonferens team affiliated to Uppsala University headed by Ulla Conti; from her team, I would like to thank specifically Maria Carlson, Maria Bäckström and Johanna Thyselius Nilsson for taking care of the website.

There are traditional ACL conference features as well - the workshops (with CoNLL-2010 as the big one), tutorials and the Student Research Workshop, the banquet (at the Uppsala Castle), the invited talks (albeit not-so-traditional this year, please come and see yourself), the Lifetime Achievement award, the business meeting and the closing session where the "conference torch" will be handed over to the Americas, as planned.

The Workshop Chairs (Pushpak Bhattacharyya and David Weir) had a hard time deciding which workshops to turn down, and tutorials had to be kept to a reasonable number, too: quite an uneasy job for the Tutorials Chairs, Lluís Màrquez and Haifeng Wang. Demos have been selected by Sandra Kübler, and exhibitions handled by Jörg Tiedemann. Publicity has been the responsibility of Koenraad de Smedt and Beata Megyesi, the local ararngements vice-chair. Students had again the opportunity to submit papers to the Student Research Workshop, organized by the SRW Chairs Nils Reiter, Seniz Demir, and Jan Raab, helped by their Faculty Advisor Tomek Strzalkowski, who also handled the application for the usual NSF grant supporting the SRW. Markéta Lopatková, the other Faculty Advisor, then centrally handled the student travel grants. Mentoring was the responsibility of Björn Gambäck and Diana McCarthy. Taking about money and the budget, the sponsoring committee has been quite successful this year by securing grants both locally and internationally: Mats Wirén, Hercules Dalianis, Christy Doran, Srinivas Bangalore, Frédérique Segond, and Steven Pulman assembled and impressive lineup of sponsors. Thanks to them, all of you can benefit from low registration fees, subsidized banquet, the conference bag, and student scholarships and prizes.

No thank-you would be complete without mentioning Priscilla Rasmussen - her experience, insight, and ability to predict the numbers and other things was extremely helpful, to say the least. The secretary-

treasurer, Graeme Hirst, has helped to reassure us whenever there was doubt or an open budgetary question. And Steven Bird, who chaired the coordinating committee (a subcommittee of the ACL and EACL executive boards) which selected the conference venue and appointed the general chair and program chairs, has been with us throughout almost two years of preparations, helping to make sure we all (read and) follow the Conference organization handbook, and address all possible problems.

Finally, a conference without your contributions (and you as participants, of course) would not happen at all. Thank you for working hard, for submitting solid work and for preparing interesting talks and posters!

Enjoy the conference.

Jan Hajič
ACL 2010 General Chair
July 2010

Preface: Program Committee Co-Chairs

Our goal this year has been to provide a broad conference program that acknowledges the very diverse field of computational linguistics and that recognizes the importance of both theoretical and empirical, data-driven research. In addition, we explicitly sought non-traditional conference papers such as surveys of important topics or emerging areas and papers that pose a challenge to the research community. A novel feature of the review process was the construction of different review forms for long and short papers and for the different types of papers, leading to a total of 20 different review forms. Not only were the review criteria for survey and challenge papers very different from the review criteria for research papers, but also the review criteria for theoretical research papers differed significantly from the review criteria for empirical research papers. Consequently, the program for the ACL 2010 conference includes a diverse set of papers, including papers in areas such as psycholinguistics and multimodal communication, as well as three survey papers and three challenge papers.

Both the number of paper submissions and attendance at ACL conferences continues to grow. Once again, the number of paper submissions to ACL 2010 broke the record set by the preceding year's conference. Discounting papers that were withdrawn or not reviewed due to failure to adhere to the specifications in the Call for Papers, there were 638 long paper submissions and 318 short paper submissions. Each submission was categorized according to topic and assigned to one of our paper tracks which were overseen by our 27 Area Chairs who selected expert reviewers to evaluate the submissions. Long paper submissions received at least three reviews and short paper submissions received at least two reviews. If there was a lack of consensus among the reviews for a paper, the reviewers then engaged in a discussion period in order to resolve disagreements. 25% of the long paper submissions and 22% of the short paper submissions were accepted for presentation at the conference. Unfortunately, even with adding a fifth parallel session, space prevented many very good papers from being accepted. As an experiment this year, authors were allowed to add an extra page of content to their final paper in order to enable them to address the comments and suggestions of their reviewers.

The conference program consists of three kinds of presentations: long 25 minute oral presentations, short 10 minute oral talks along with a subsequent poster presentation, and solely poster presentations. While the conference proceedings differentiates papers according to whether they are long or short papers, there is no distinction in the proceedings with respect to the mode of presentation.

We are delighted to have two invited speakers who will give what promise to be exciting plenary lectures. Andrei Broder (Vice-president, Yahoo! Research) will discuss the emerging field of computational advertising. Zenzi Griffin (Professor, University of Texas) will discuss language processing in interpersonal interactions. In addition, the recipient of the ACL Lifetime Achievement Award will present a plenary lecture on the second day of the conference.

As is traditional, there will be an award for the best long paper, the best long paper by a student, and the best short paper. Reviewers were asked to indicate whether a paper might merit a best paper prize. Area chairs then reviewed suggested papers and nominated deserving papers for further consideration. Two Best Paper Prize Committees were formed, one for long papers and one for short papers; each committee consisted of 5-6 senior researchers. The committees, along with the Program Chairs, then selected the award recipients. The Best Long Paper Prize will be presented at the plenary final session of the conference on Wednesday afternoon, and the recipient will present his or her paper during this session. The prize for Best Long Paper by a Student and the prize for the Best Short Paper will be presented at the oral talks given by the recipients during one of the regular paper sessions.

As usual, there are many individuals to thank for their contributions to the conference program. Most notably, we thank our 27 Area Chairs who did a superb job of overseeing the review of papers in their domain, the members of the Best Paper Committees, and the over 600 reviewers who in almost all cases

provided detailed, comprehensive reviews. We also owe a tremendous debt of gratitude to Rich Gerber; his START system made the review and scheduling process manageable. Moreover, his quick response to questions and requested modifications were very much appreciated. We want to thank Jason Eisner for his web site that provides invaluable advice on "How to Serve as a Program Chair of a Conference". We were very fortunate to work with wonderful organizers: Jan Hajic as General Chair, Joakim Nivre as Local Arrangements Chair, Jing-Shin Chang and Philipp Koehn as Publications Co-Chairs, and the ACL 2010 Coordinating Committee. We also thank last year's program chairs, Jian Su and Jan Wiebe, for their advice and responses to our questions.

But the ACL conference is more than just the main technical presentations. We would like to acknowledge the efforts of Seniz Demir, Jan Raab, and Nils Reiter, along with the faculty advisers Marketa Lopatkova and Tomek Strzalkowski, for organizing the Student Research Workshop. We would also like to acknowledge the work of Lluis Marquez and Haifeng Wang for soliciting an excellent set of pre-conference tutorials, Pushpak Bhattacharyia and David Weir for compiling a set of very interesting post-conference workshops, Sandra Kubler for assembling informative system demonstrations, and Jörg Tiedemann for handling exhibits.

We hope that you enjoy the conference!

ACL 2010 Program Co-Chairs
Sandra Carberry, University of Delaware
Stephen Clark, University of Cambridge

Organizing Committee

General Chair:

Jan Hajič, Charles University in Prague, Czech Republic

Local Arrangements Chair:

Joakim Nivre, Uppsala University, Sweden

Program Chairs:

Sandra Carberry, University of Delaware, USA
Stephen Clark, University of Cambridge, United Kingdom

Local Arrangements Committee:

Beata Megyesi, Uppsala University, Sweden (vice-chair)
Anna Sågvall Hein, Uppsala University, Sweden
Rolf Carlson, KTH, Stockholm, Sweden
Mats Dahllöf, Uppsala University, Sweden
Marco Kuhlmann, Uppsala University, Sweden
Mattias Nilsson, Uppsala University, Sweden
Markus Saers, Uppsala University, Sweden
Per Starbäck, Uppsala University, Sweden
Oscar Täckström, Uppsala University, Sweden
Jörg Tiedemann, Uppsala University, Sweden
Reut Tsarfaty, Uppsala University, Sweden

Local Organizing Secretariat (Akademikonferens, Uppsala, Sweden):

Ulla Conti (head)
Maria Carlson, Maria Bäckström, Johanna Tyselius Nilsson (webmasters)

Publicity Chairs:

Koenraad de Smedt, University of Bergen, Norway
Beata Megyesi, Uppsala University, Sweden

Exhibits Chair:

Jörg Tiedemann, Uppsala University, Sweden

Sponsorship Chairs:

Stephen Pulman, University of Oxford, United Kingdom
Frédérique Segond, Xerox Research Centre Europe, France
Srinivas Bangalore, AT&T Labs-Research, USA
Christy Doran, MITRE, USA
Hercules Dalianis, Stockholm University, Sweden

Mats Wirén, Stockholm University, Sweden

Workshop Chairs:

David Weir, University of Sussex, United Kingdom
Pushpak Bhattacharyya, IIT Mumbai, India

Tutorial Chairs:

Lluís Màrquez, Technical University of Catalonia, Spain
Haifeng Wang, Baidu, China

Publications Chairs:

Jing-Shin Chang, National Chi Nan University, Taiwan
Philipp Koehn, University of Edinburgh, United Kingdom

System Demonstrations Chair:

Sandra Kübler, Indiana University, USA

Student Research Workshop Chairs:

Nils Reiter, University of Heidelberg, Germany
Seniz Demir, University of Delaware, USA
Jan Raab, Charles University in Prague, Czech Republic
Tomek Strzalkowski, University at Albany – SUNY, USA (Faculty advisor)
Markéta Lopatková, Charles University in Prague (Faculty advisor)

Mentoring Service Chairs:

Björn Gambäck, SICS, Sweden
Diana McCarthy, Lexical Computing Ltd., United Kingdom

Registration:

Priscilla Rasmussen, Association for Computational Linguistics (ACL), USA

ACL Conference Coordinating Committee Chair for 2010:

Steven Bird, University of Melbourne, Australia

Mentoring Service Chairs:

Björn Gambäck, SICS, Sweden
Diana McCarthy, Lexical Computing Ltd., United Kingdom

Mentoring Committee:

Chris Brew, The Ohio State University, USA
John Carroll, University of Sussex, United Kingdom
Ann Copestake, University of Cambridge, United Kingdom
Robert Daland,, University of California at Los Angeles, USA
Roger Evans, University of Brighton, United Kingdom
Debora Field, University of Sheffield, United Kingdom
Louise Guthrie, Oxford Internet Institute, United Kingdom
Graeme Hirst, University of Toronto, Canada
Nancy Ide, Vassar College, USA
Kemal Oflazer, Carnegie Mellon University – Qatar
Dan Roth, University of Illinois at Urbana-Champaign, USA
Donia Scott, University of Sussex, United Kingdom
Mark Steedman, University of Edinburgh, United Kingdom
Mark Stevenson, University of Sheffield, United Kingdom
Antal van den Bosch, Tilburg University, Germany
Nick Webb, University at Albany – SUNY, USA
Bonnie Webber, University of Edinburgh, United Kingdom
David Weir, University of Sussex, United Kingdom
Richard Wicentowski, Swarthmore College, USA
Graham Wilcock, University of Helsinki, Finland

Program Committee

Program Chairs:

Sandra Carberry, University of Delaware, USA
Stephen Clark, University of Cambridge, United Kingdom

Area Chairs:

Tim Baldwin (University of Melbourne, Australia)
Phil Blunsom (University of Oxford, UK)
Kalina Bontcheva (University of Sheffield, UK)
Johan Bos (University of Rome - La Sapienza, Italy)
Claire Cardie (Cornell University, United States)
Walter Daelemans (University of Antwerp, Belgium)
Rob Gaizauskas (University of Sheffield, UK)
Keith Hall (Google Research - Zurich, Switzerland)
Julia Hirschberg (Columbia University, United States)
Nancy Ide (Vassar College, United States)
Michael Johnston (AT&T Labs, United States)
Roger Levy (University of California - San Diego, United States)
Hang Li (Microsoft Research Asia, China)
Chin-Yew Lin (Microsoft Research Asia, China)
Yusuke Miyao (University of Tokyo, Japan)
Roberto Navigli (University of Rome - La Sapienza, Italy)
Ani Nenkova (University of Pennsylvania, United States)
Jon Oberlander (University of Edinburgh, UK)
Chris Quirk (Microsoft Research, United States)
Stuart Shieber (Harvard University, United States)
Khalil Sima'an (University of Amsterdam, The Netherlands)
Richard Sproat (Oregon Health and Science University, United States)
Matthew Stone (Rutgers University, United States)
Jun'ichi Tsujii (University of Tokyo and University of Manchester, Japan and UK)
Bonnie Webber (University of Edinburgh, UK)
Theresa Wilson (University of Edinburgh, UK)
ChengXiang Zhai (University of Illinois at Urbana-Champaign, United States)

Best Paper Committee:

David Chiang (Information Sciences Institute, United States)
James R. Curran (University of Sydney, Australia)
Walter Daelemans (University of Antwerp, Belgium)
Nancy Ide (Vassar College, United States)
Ann Copestake (University of Cambridge, UK)
Julia Hirschberg (Columbia University, United States)
Mark-Jan Nederhof (University of St Andrews, UK)
Stefan Riezler (University of Heidelberg, Germany)
Brian Roark (Oregon Health and Science University, United States)
Rada Mihalcea (University of North Texas, United States)

Program Committee:

Ahmed Abbasi, Anne Abeille, Eugene Agichtein, Eneko Agirre, David Ahn, Beatrice Alex, Enrique Alfonseca, Afra Alishahi, Alina Andreevskaia, Doug Appelt, Abhishek Arun, Nicholas Asher, Aiti Aw

Bogdan Babych, Nguyen Bach, Collin Baker, Tim Baldwin, Carmen Banea, Srinivas Bangalore, Colin Bannard, Ken Barker, Regina Barzilay, Roberto Basili, Anton Batliner, Tilman Becker, Nuria Bel, Anja Belz, Luisa Bentivogli, Sabine Bergler, Raffaella Bernardi, Tim Bickmore, Klinton Bicknell, Dan Bikel, Steven Bird, Rens Bod, Francis Bond, Stefan Bordag, Antal van den Bosch, Gosse Bouma, Jordan Boyd-Graber, S.R.K. Branavan, Eric Breck, Chris Brew, Ted Briscoe, Samuel Brody, Sabine Buchholz, Paul Buitelaar, Razvan Bunescu, Harry Bunt, Aljoscha Burchardt, Stephan Busemann, Miriam Butt, Bill Byrne

Michael Cafarella, Aoife Cahill, Chris Callison-Burch, Diego Calvanese, Nicoletta Calzolari, Nicola Cancedda, Yunbo Cao, Giuseppe Carenini, Michael Carl, Bob Carpenter, Marine Carpuat, Xavier Carreras, John Carroll, Ben Carterette, Diamantino Caseiro, Dan Cer, Joyce Chai, Yee Seng Chan, Jason Chang, Pi-Chuan Chang, Ciprian Chelba, Aitao Chen, Hsin-Hsi Chen, Stanley Chen, Wenliang Chen, Colin Cherry, David Chiang, Key-Sun Choi, Yejin Choi, Tat-Seng Chua, Kenneth Church, Massimiliano Ciaramita, Christopher Cieri, Philipp Cimiano, Alex Clark, Peter Clark, James Clarke, Andrew Clegg, Eric de la Clergerie, Aaron Cohen, Kevin Cohen, Shay Cohen, Trevor Cohn, Nigel Collier, Michael Collins, Kevyn Collins-Thompson, Gao Cong, Paul Cook, Anna Corazza, Marta Ruiz Costa-Jussa, Josep Crego, Mathias Creutz, Dan Cristea, Montse Cuadros, Hang Cui, James Curran

Robert Damper, Dipanjan Das, Hal Daume, Thierry Declerck, Vera Demberg, Steve DeNeefe, John DeNero, Tejaswini Deoskar, Ann Devitt, Mona Diab, Bill Dolan, Christy Doran, Doug Downey, Mark Dredze, Markus Dreyer, Greg Druck, Amit Dubey, Kevin Duh, Benjamin van Durme, Chris Dyer

Markus Egg, Koji Eguchi, Jason Eisner, Michael Elhadad, Ralf Engbert, Tomaz Erjavec, Katrin Erk, Gunes Erkan, Gülşen Eryiğit, Maxine Eskenazi, Andrea Esuli, Oren Etzioni

Hui Fang, Marcello Federico, Anna Feldman, Ronen Feldman, Christiane Fellbaum, Raquel Fernandez, Raul Fernandez, Tim Fernando, Jenny Finkel, Dan Flickinger, Radu Florian, Juliane Fluck, Kate Forbes-Riley, Mary Ellen Foster, Nissim Francez, Anette Frank, Stefan Frank, Stella Frank, Alex Fraser, Dayne Freitag, Pascale Fung, Sadaoki Furui, Dina Demner Fushman

Evgeniy Gabrilovich, Michael Galley, Michael Gamon, Jianfeng Gao, Claire Gardent, Nikesh Garera, Josef van Genabith, Dmitriy Genzel, Kallirroi Georgila, Daniel Gildea, Alastair Gill, Jesus Gimenez, Kevin Gimpel, Roxana Girju, Adria de Gispert, Claudio Giuliano, Sharon Goldwater, Gevieve Gorrell, Agustin Gravano, Stephan Greene, Mark Greenwood, Gregory Grefenstette, Ralph Grishman, Philippe de Groote, Claire Grover, Joakim Gustafson

Barry Haddow, Gholamreza Haffari, Udo Hahn, Dilek Hakkani-Tur, Harald Hammarstrom, Siegfried Handschuh, Sanda Harabagiu, Henk Harkema, Donna Harman, Saša Hasan, Chikara Hashimoto, Hany Hassan, Xiaodong He, Peter Heeman, James Henderson, Mark Hepple, Ulf Hermjakob, Andrew Hickl, Ryuichiro Higashinaka, Erhard Hinrichs, Lynette Hirschman, Graeme Hirst, Julia Hockenmaier, Beth Ann Hockey, Veronique Hoste, Yunnhua Hu, Chu-Ren Huang, Jimmy Huang, Liang Huang, Minlie Huang, Xuanjing Huang, Sarmad Hussain, Gerhard van Huyssteen, Rebecca Hwa

Nancy Ide, Diana Inkpen

Florian Jaeger, Gerhard Jaeger, Martin Jansche, Jing Jiang, Wenbin Jiang, Richard Johansson, Howard Johnson, Mark Johnson, Rie Johnson, Michael Johnston, Aravind Joshi

Damianos Karakos, Nikiforos Karamanis, Lauri Karttunen, Daisuke Kawahara, Andrew Kehler, Frank Keller, Rodger Kibble, Adam Kilgarriff, Jin-Dong Kim, Su Nam Kim, Tracy Holloway

King, Kevin Knight, Alistair Knott, Philipp Koehn, Rob Koeling, Oskar Kohonen, Alexander Koller, Greg Kondrak, Moshe Koppel, Valia Kordoni, Anna Korhonen, Andras Kornai, Wessel Kraaij, Emiel Krahmer, Ivana Kruijff-Korbayova, Udo Kruschwitz, Lun-Wei Ku, Sandra Kuebler, Marco Kuhlmann, Roland Kuhn, Shankar Kumar, Mikko Kurimo, Oren Kurland, Sadao Kurohashi, Tom Kwiatkowski, Kui-Lam Kwok, Olivia Kwong

Krista Lagus, Patrik Lambert, Philippe Langlais, Mirella Lapata, Alex Lascarides, Alberto Lavelli, Victor Lavrenko, Gary Geunbae Lee, Yoong Keok Lee, Gregor Leusch, Gina-Anne Levow, Roger Levy, David Lewis, Chi-Ho Li, Haizhou Li, Maggie Wenjie Li, Mu Li, Wei Li, Xiao Li, Zhifei Li, Percy Liang, Chuan-Jie Lin, Ken Litkowski, Diane Litman, Bing Liu, Qun Liu, Ting Liu, Yang Liu, Yupeng Liu, Adam Lopez, Ramon Lopez-Cozar, Daniel Lopresti, Annie Louis, Bernd Ludwig

Klaus Macherey, Wolfgang Macherey, Rob Malouf, Suresh Manandhar, Daniel Marcu, Mitch Marcus, Katja Markert, Erwin Marsi, Andre Martins, David Martínez, Sameer Maskey, Yuji Matsumoto, Takuya Matsuzaki, Evgeny Matusov, Arne Mauser, Mike Maxwell, Diana Maynard, Diana McCarthy, David McClosky, Ryan McDonald, Kathleen McKeown, Qiaozhu Mei, Dan Melamed, Arul Menezes, Helen Meng, Paola Merlo, Donald Metzler, Haitao Mi, Jens Michaelis, Rada Mihalcea, Eleni Miltsakaki, David Mimno, Michael Minock, Teruko Mitamura, Bernd Moebius, Marie-Francine Moens, Saif Mohammad, Christian Monson, Christof Monz, Bob Moore, Michael Moortgat, Roser Morante, Louis-Philippe Morency, Tatsunori Mori, Glyn Morrill, Alessandro Moschitti, Larry Moss, Lluís Màrquez, Tony Mullen, Gabriel Murray, Sung-Hyon Myaeng

Tetsuji Nakagawa, Hiromi Nakaiwa, Yukiko Nakano, Tahira Naseem, Vivi Nastase, Tetsuya Nasukawa, Mark-Jan Nederhof, Rani Nelken, Goran Nenadic, Ani Nenkova, Günter Neumann, Hermann Ney, Hwee Tou Ng, Vincent Ng, Patrick Nguyen, Jeremy Nicholson, Jian-Yun Nie, Zaiqing Nie, Takashi Ninomiya, Gertjan van Noord, Scott Nowson

Jon Oberlander, Franz Och, Stephan Oepen, Kemal Oflazer, Naoaki Okazaki, Manabu Okumura, Constantin Orasan, Miles Osborne, Diarmuid OSeaghdha

Sebastian Pado, Tim Paek, Bo Pang, Patrick Pantel, Soo-Min Pantel, Cecile Paris, Marius Pasca, Rebecca Passonneau, Adam Pauls, Guy De Pauw, Gerald Penn, Fernando Pereira, John Pestian, Kay Peterson, Slav Petrov, Michael Picheny, Roberto Pieraccini, Manfred Pinkal, Stelios Piperidis, Emily Pitler, Paul Piwek, Massimo Poesio, Simone Paolo Ponzetto, Hoifung Poon, Ana-Maria Popescu, Andrei Popescu-Belis, Matt Post, David Powers, John Prager, Rashmi Prasad, Stephen Pulman, Vasin Punyakanok, Matthew Purver, James Pustejovsky, Sampo Pyysalo

Stephan Raaijmakers, Filip Radlinski, Bhuvana Ramabhadran, Owen Rambow, Allan Ramsay, Delip Rao, Reinhard Rapp, Ari Rappoport, Emmanuel Rayner, Dietrich Rebholz-Schuhmann, Flo Reeder, Erik Reichle, Ehud Reiter, David Reitter, Steve Renals, Christian Retore, Giuseppe Riccardi, Sebastian Riedel, German Rigau, Ellen Riloff, Hae-Chang Rim, Laura Rimell, Brian Roark, James Rogers, Doug Roland, Laurent Romary, Andrew Rosenberg, Paolo Rosso, Dan Roth, Shourya Roy, Patrick Ruch

Kenji Sagae, William Sakas, Tapio Salakoski, Greg Sanders, Rajeev Sangal, Murat Saraclar, Giorgio Satta, Roser Sauri, Frank Schilder, David Schlangen, Helmut Schmid, Hinrich Schuetze, William Schuler, Rolf Schwitter, Donia Scott, Yohei Seki, Satoshi Sekine, Violeta Seretan, Burr Settles, Chung-chieh Shan, Vijay Shanker, Libin Shen, Stuart Shieber, Kiyoaki Shirai, Mario Silva, David Smith, Jason Smith, Nathaniel Smith, Noah Smith, Ben Snyder, Stephen Soderland, Hagen Soltau, Swapna Somasundaran, Young-In Song, Abdelhadi Soudi, Lucia Specia, Sebastian Spiegler, Michael Spivey, Caroline Sporleder, Richard Sproat, Rohini Srihari, Padmini Srinivasan, Ed Stabler, Manfred Stede, Mark Steedman, Ralf Steinberger, Amanda Stent, Mark Stevenson, Matthew Stone, Veselin Stoyanov, Carlo Strapparava, Michael Strube, Patrick Sturt, Jian Su, L Venkata Subramaniam, Mihai Surdeanu, Marc Swerts, Stan Szpakowicz

Hiroya Takamura, David Talbot, Kumiko Tanaka-Ishii, Jie Tang, Ben Taskar, Michael Tepper, Stefan Thater, Mariet Theune, Christoph Tillmann, Ivan Titov, Take Tokunaga, Kentaro Torisawa,

Kristina Toutanova, Roy Tromble, Vivian Tsang, Reut Tsarfaty, Yoshimasa Tsuruoka, Gokhan Tur, Peter Turney

Lucy Vanderwende, Sebastian Varges, Vasudeva Varma, Shravan Vasishth, Tony Veale, Paola Velardi, Henriette Viethen, Marc Vilain, David Vilar, Sami Virpioja, Andreas Vlachos, Stephan Vogel, Piek Vossen

Sabine Schulte im Walde, Hanna Wallach, Stephen Wan, Xiaojun Wan, Haifeng Wang, Wei Wang, Taro Watanabe, Andy Way, Bonnie Webber, David Weir, Ralph Weischedel, Ben Wellner, Edward Whittaker, Janyce Wiebe, John Wilbur, Theresa Wilson, Shuly Wintner, Yuk Wah Wong, Dekai Wu

Fei Xia, Deyi Xiong, Peng Xu, Nianwen Xue

Mei Yang, Roman Yangarber, Patrick Ye, Lars Yencken, Zhang Yi, Scott Wen-tau Yih

Richard Zens, Luke Zettlemoyer, Cheng Zhai, Hao Zhang, Yi Zhang, Yue Zhang, Zhu Zhang, Jing Zheng, Jun Zhu, Andreas Zollmann, Willem (Jelle) Zuidema, Ingrid Zukerman

Table of Contents

Efficient Third-order Dependency Parsers

Terry Koo and Michael Collins
MIT CSAIL, Cambridge, MA, 02139, USA
{maestro,mcollins}@csail.mit.edu

Abstract

We present algorithms for higher-order dependency parsing that are "third-order" in the sense that they can evaluate substructures containing three dependencies, and "efficient" in the sense that they require only $O(n^4)$ time. Importantly, our new parsers can utilize both sibling-style and grandchild-style interactions. We evaluate our parsers on the Penn Treebank and Prague Dependency Treebank, achieving unlabeled attachment scores of 93.04% and 87.38%, respectively.

1 Introduction

Dependency grammar has proven to be a very useful syntactic formalism, due in no small part to the development of efficient parsing algorithms (Eisner, 2000; McDonald et al., 2005b; McDonald and Pereira, 2006; Carreras, 2007), which can be leveraged for a wide variety of learning methods, such as feature-rich discriminative models (Lafferty et al., 2001; Collins, 2002; Taskar et al., 2003). These parsing algorithms share an important characteristic: they *factor* dependency trees into sets of *parts* that have limited interactions. By exploiting the additional constraints arising from the factorization, maximizations or summations over the set of possible dependency trees can be performed efficiently and exactly.

A crucial limitation of factored parsing algorithms is that the associated parts are typically quite small, losing much of the contextual information within the dependency tree. For the purposes of improving parsing performance, it is desirable to increase the size and variety of the parts used by the factorization.[1] At the same time, the need for more expressive factorizations must be balanced against any resulting increase in the computational cost of the parsing algorithm. Consequently, recent work in dependency parsing has been restricted to applications of second-order parsers, the most powerful of which (Carreras, 2007) requires $O(n^4)$ time and $O(n^3)$ space, while being limited to second-order parts.

In this paper, we present new third-order parsing algorithms that increase both the size and variety of the parts participating in the factorization, while simultaneously maintaining computational requirements of $O(n^4)$ time and $O(n^3)$ space. We evaluate our parsers on the Penn WSJ Treebank (Marcus et al., 1993) and Prague Dependency Treebank (Hajič et al., 2001), achieving unlabeled attachment scores of 93.04% and 87.38%. In summary, we make three main contributions:

1. Efficient new third-order parsing algorithms.
2. Empirical evaluations of these parsers.
3. A free distribution of our implementation.[2]

The remainder of this paper is divided as follows: Sections 2 and 3 give background, Sections 4 and 5 describe our new parsing algorithms, Section 6 discusses related work, Section 7 presents our experimental results, and Section 8 concludes.

2 Dependency parsing

In dependency grammar, syntactic relationships are represented as head-modifier dependencies: directed arcs between a *head*, which is the more "essential" word in the relationship, and a *modifier*, which supplements the meaning of the head. For example, Figure 1 contains a dependency between the verb "report" (the head) and its object "sales" (the modifier). A complete analysis of a sentence is given by a dependency tree: a set of dependencies that forms a rooted, directed tree spanning the words of the sentence. Every dependency tree is rooted at a special "*" token, allowing the

[1] For examples of how performance varies with the degree of the parser's factorization see, e.g., McDonald and Pereira (2006, Tables 1 and 2), Carreras (2007, Table 2), Koo et al. (2008, Tables 2 and 4), or Suzuki et al. (2009, Tables 3–6).

[2] http://groups.csail.mit.edu/nlp/dpo3/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1–11,
Uppsala, Sweden, 11-16 July 2010.

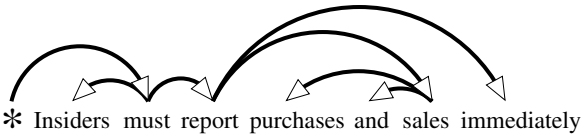


Figure 1: An example dependency structure.

selection of the sentential head to be modeled as if it were a dependency.

For a sentence $\boldsymbol{x}$, we define dependency parsing as a search for the highest-scoring analysis of $\boldsymbol{x}$:

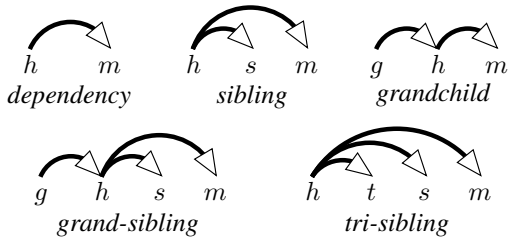
$$y^*(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}(\boldsymbol{x})} \textsc{Score}(\boldsymbol{x}, y) \quad (1)$$

Here, $\mathcal{Y}(\boldsymbol{x})$ is the set of all trees compatible with $\boldsymbol{x}$ and $\textsc{Score}(\boldsymbol{x}, y)$ evaluates the event that tree y is the analysis of sentence $\boldsymbol{x}$. Since the cardinality of $\mathcal{Y}(\boldsymbol{x})$ grows exponentially with the length of the sentence, directly solving Eq. 1 is impractical. A common strategy, and one which forms the focus of this paper, is to *factor* each dependency tree into small *parts*, which can be scored in isolation. Factored parsing can be formalized as follows:

$$\textsc{Score}(\boldsymbol{x}, y) = \sum_{p \in y} \textsc{ScorePart}(\boldsymbol{x}, p)$$

That is, we treat the dependency tree y as a set of parts p, each of which makes a separate contribution to the score of y. For certain factorizations, efficient parsing algorithms exist for solving Eq. 1.

We define the *order* of a part according to the number of dependencies it contains, with analogous terminology for factorizations and parsing algorithms. In the remainder of this paper, we focus on factorizations utilizing the following parts:



Specifically, Sections 4.1, 4.2, and 4.3 describe parsers that, respectively, factor trees into grandchild parts, grand-sibling parts, and a mixture of grand-sibling and tri-sibling parts.

3 Existing parsing algorithms

Our new third-order dependency parsers build on ideas from existing parsing algorithms. In this section, we provide background on two relevant parsers from previous work.

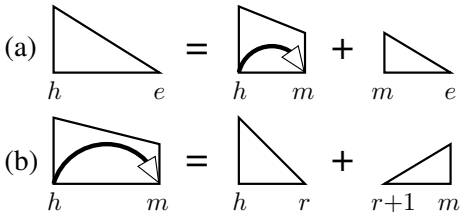


Figure 2: The dynamic-programming structures and derivations of the Eisner (2000) algorithm. Complete spans are depicted as triangles and incomplete spans as trapezoids. For brevity, we elide the symmetric right-headed versions.

3.1 First-order factorization

The first type of parser we describe uses a "first-order" factorization, which decomposes a dependency tree into its individual dependencies. Eisner (2000) introduced a widely-used dynamic-programming algorithm for first-order parsing; as it is the basis for many parsers, including our new algorithms, we summarize its design here.

The Eisner (2000) algorithm is based on two interrelated types of dynamic-programming structures: *complete* spans, which consist of a head-word and its descendents on one side, and *incomplete* spans, which consist of a dependency and the region between the head and modifier.

Formally, we denote a complete span as $C_{h,e}$ where h and e are the indices of the span's head-word and endpoint. An incomplete span is denoted as $I_{h,m}$ where h and m are the index of the head and modifier of a dependency. Intuitively, a complete span represents a "half-constituent" headed by h, whereas an incomplete span is only a partial half-constituent, since the constituent can be extended by adding more modifiers to m.

Each type of span is created by recursively combining two smaller, adjacent spans; the constructions are specified graphically in Figure 2. An incomplete span is constructed from a pair of complete spans, indicating the division of the range $[h, m]$ into constituents headed by h and m. A complete span is created by "completing" an incomplete span with the other half of m's constituent. The point of concatenation in each construction—m in Figure 2(a) or r in Figure 2(b)—is the *split point*, a free index that must be enumerated to find the optimal construction.

In order to parse a sentence $\boldsymbol{x}$, it suffices to find optimal constructions for all complete and incomplete spans defined on $\boldsymbol{x}$. This can be

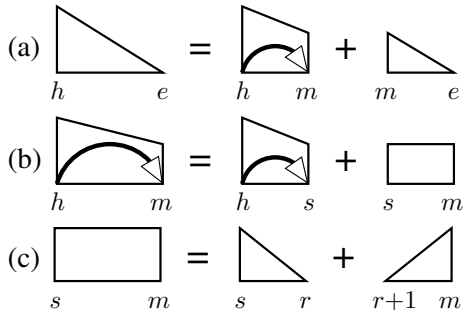


Figure 3: The dynamic-programming structures and derivations of the second-order sibling parser; sibling spans are depicted as boxes. For brevity, we elide the right-headed versions.

accomplished by adapting standard chart-parsing techniques (Cocke and Schwartz, 1970; Younger, 1967; Kasami, 1965) to the recursive derivations defined in Figure 2. Since each derivation is defined by two fixed indices (the boundaries of the span) and a third free index (the split point), the parsing algorithm requires $O(n^3)$ time and $O(n^2)$ space (Eisner, 1996; McAllester, 1999).

3.2 Second-order sibling factorization

As remarked by Eisner (1996) and McDonald and Pereira (2006), it is possible to rearrange the dynamic-programming structures to conform to an improved factorization that decomposes each tree into *sibling* parts—pairs of dependencies with a shared head. Specifically, a sibling part consists of a triple of indices (h, m, s) where (h, m) and (h, s) are dependencies, and where s and m are successive modifiers to the same side of h.

In order to parse this factorization, the second-order parser introduces a third type of dynamic-programming structure: *sibling* spans, which represent the region between successive modifiers of some head. Formally, we denote a sibling span as $S_{s,m}$ where s and m are a pair of modifiers involved in a sibling relationship. Modified versions of sibling spans will play an important role in the new parsing algorithms described in Section 4.

Figure 3 provides a graphical specification of the second-order parsing algorithm. Note that incomplete spans are constructed in a new way: the second-order parser combines a smaller incomplete span, representing the next-innermost dependency, with a sibling span that covers the region between the two modifiers. Sibling parts (h, m, s) can thus be obtained from Figure 3(b). Despite the use of second-order parts, each derivation is still defined by a span and split point, so the parser requires $O(n^3)$ time and $O(n^2)$ space.

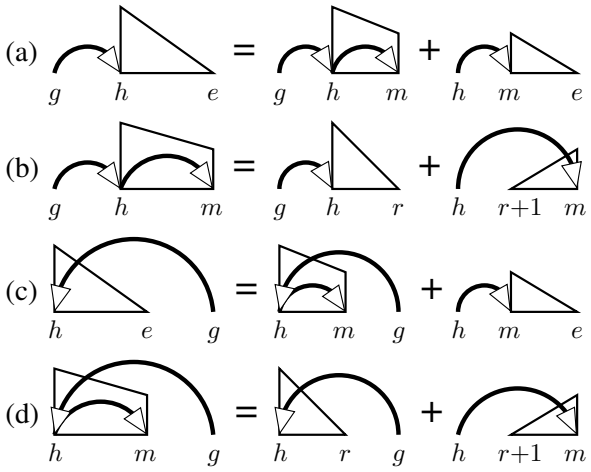


Figure 4: The dynamic-programming structures and derivations of Model 0. For brevity, we elide the right-headed versions. Note that (c) and (d) differ from (a) and (b) only in the position of g.

4 New third-order parsing algorithms

In this section we describe our new third-order dependency parsing algorithms. Our overall method is characterized by the augmentation of each span with a "grandparent" index: an index external to the span whose role will be made clear below. This section presents three parsing algorithms based on this idea: Model 0, a second-order parser, and Models 1 and 2, which are third-order parsers.

4.1 Model 0: all grandchildren

The first parser, Model 0, factors each dependency tree into a set of *grandchild* parts—pairs of dependencies connected head-to-tail. Specifically, a grandchild part is a triple of indices (g, h, m) where (g, h) and (h, m) are dependencies.[3]

In order to parse this factorization, we augment both complete and incomplete spans with grandparent indices; for brevity, we refer to these augmented structures as *g-spans*. Formally, we denote a complete g-span as $C^g_{h,e}$, where $C_{h,e}$ is a normal complete span and g is an index lying outside the range $[h, e]$, with the implication that (g, h) is a dependency. Incomplete g-spans are defined analogously and are denoted as $I^g_{h,m}$.

Figure 4 depicts complete and incomplete g-spans and provides a graphical specification of the

[3]The Carreras (2007) parser also uses grandchild parts but only in restricted cases; see Section 6 for details.

OPTIMIZEALLSPANS($\boldsymbol{x}$)

1. $\forall g, i \quad C^g_{i,i} = 0$ ◁ *base case*
2. **for** $w = 1 \ldots (n-1)$ ◁ *span width*
3. **for** $i = 1 \ldots (n-w)$ ◁ *span start index*
4. $j = i + w$ ◁ *span end index*
5. **for** $g < i$ **or** $g > j$ ◁ *grandparent index*
6. $I^g_{i,j} = \max_{i \le r < j} \{C^g_{i,r} + C^i_{j,r+1}\} +$ SCOREG($\boldsymbol{x}, g, i, j$)
7. $I^g_{j,i} = \max_{i \le r < j} \{C^g_{j,r+1} + C^j_{i,r}\} +$ SCOREG($\boldsymbol{x}, g, j, i$)
8. $C^g_{i,j} = \max_{i < m \le j} \{I^g_{i,m} + C^i_{m,j}\}$
9. $C^g_{j,i} = \max_{i \le m < j} \{I^g_{j,m} + C^j_{m,i}\}$
10. **endfor**
11. **endfor**
12. **endfor**

Figure 5: A bottom-up chart parser for Model 0. SCOREG is the scoring function for grandchild parts. We use the g-span identities as shorthand for their chart entries (e.g., $I^g_{i,j}$ refers to the entry containing the maximum score of that g-span).

Model 0 dynamic-programming algorithm. The algorithm resembles the first-order parser, except that every recursive construction must also set the grandparent indices of the smaller g-spans; fortunately, this can be done deterministically in all cases. For example, Figure 4(a) depicts the decomposition of $C^g_{h,e}$ into an incomplete half and a complete half. The grandparent of the incomplete half is copied from $C^g_{h,e}$ while the grandparent of the complete half is set to h, the head of m as defined by the construction. Clearly, grandchild parts (g, h, m) can be read off of the incomplete g-spans in Figure 4(b,d). Moreover, since each derivation copies the grandparent index g into successively smaller g-spans, grandchild parts will be produced for *all* grandchildren of g.

Model 0 can be parsed by adapting standard top-down or bottom-up chart parsing techniques. For concreteness, Figure 5 provides a pseudocode sketch of a bottom-up chart parser for Model 0; although the sketch omits many details, it suffices for the purposes of illustration. The algorithm progresses from small widths to large in the usual manner, but after defining the endpoints (i, j) there is an additional loop that enumerates all possible grandparents. Since each derivation is defined by three fixed indices (the g-span) and one free index (the split point), the complexity of the algorithm is $O(n^4)$ time and $O(n^3)$ space.

Note that the grandparent indices cause each g-span to have non-contiguous structure. For example, in Figure 4(a) the words between g and h will be controlled by some other g-span. Due to these discontinuities, the correctness of the Model 0 dynamic-programming algorithm may not be immediately obvious. While a full proof of correctness is beyond the scope of this paper, we note that each structure on the right-hand side of Figure 4 lies completely within the structure on the left-hand side. This nesting of structures implies, in turn, that the usual properties required to ensure the correctness of dynamic programming hold.

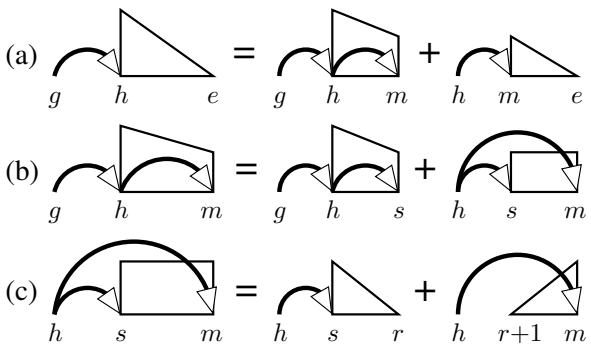


Figure 6: The dynamic-programming structures and derivations of Model 1. Right-headed and right-grandparented versions are omitted.

4.2 Model 1: all grand-siblings

We now describe our first third-order parsing algorithm. Model 1 decomposes each tree into a set of *grand-sibling* parts—combinations of sibling parts and grandchild parts. Specifically, a grand-sibling is a 4-tuple of indices (g, h, m, s) where (h, m, s) is a sibling part and (g, h, m) and (g, h, s) are grandchild parts. For example, in Figure 1, the words "must," "report," "sales," and "immediately" form a grand-sibling part.

In order to parse this factorization, we introduce sibling g-spans $S^h_{m,s}$, which are composed of a normal sibling span $S_{m,s}$ and an external index h, with the implication that (h, m, s) forms a valid sibling part. Figure 6 provides a graphical specification of the dynamic-programming algorithm for Model 1. The overall structure of the algorithm resembles the second-order sibling parser, with the addition of grandparent indices; as in Model 0, the grandparent indices can be set deterministically in all cases. Note that the sibling g-spans are crucial: they allow grand-sibling parts (g, h, m, s) to be read off of Figure 6(b), while simultaneously propagating grandparent indices to smaller g-spans.

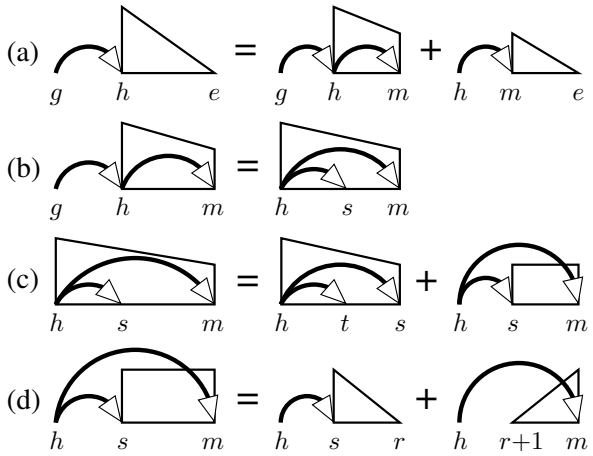


Figure 7: The dynamic-programming structures and derivations of Model 2. Right-headed and right-grandparented versions are omitted.

Like Model 0, Model 1 can be parsed via adaptations of standard chart-parsing techniques; we omit the details for brevity. Despite the move to third-order parts, each derivation is still defined by a g-span and a split point, so that parsing requires only $O(n^4)$ time and $O(n^3)$ space.

4.3 Model 2: grand-siblings and tri-siblings

Higher-order parsing algorithms have been proposed which extend the second-order sibling factorization to parts containing multiple siblings (McDonald and Pereira, 2006, also see Section 6 for discussion). In this section, we show how our g-span-based techniques can be combined with a third-order sibling parser, resulting in a parser that captures both grand-sibling parts and *tri-sibling* parts—4-tuples of indices (h, m, s, t) such that both (h, m, s) and (h, s, t) are sibling parts.

In order to parse this factorization, we introduce a new type of dynamic-programming structure: sibling-augmented spans, or *s-spans*. Formally, we denote an incomplete s-span as $I_{h,m,s}$ where $I_{h,m}$ is a normal incomplete span and s is an index lying in the strict interior of the range $[h, m]$, such that (h, m, s) forms a valid sibling part.

Figure 7 provides a graphical specification of the Model 2 parsing algorithm. An incomplete s-span is constructed by combining a smaller incomplete s-span, representing the next-innermost pair of modifiers, with a sibling g-span, covering the region between the outer two modifiers. As in Model 1, sibling g-spans are crucial for propagating grandparent indices, while allowing the recovery of tri-sibling parts (h, m, s, t). Figure 7(b) shows how an incomplete s-span can be converted into an incomplete g-span by exchanging the internal sibling index for an external grandparent index; in the process, grand-sibling parts (g, h, m, s) are enumerated. Since every derivation is defined by an augmented span and a split point, Model 2 can be parsed in $O(n^4)$ time and $O(n^3)$ space.

It should be noted that unlike Model 1, Model 2 produces grand-sibling parts only for the outermost pair of grandchildren,[4] similar to the behavior of the Carreras (2007) parser. In fact, the resemblance is more than passing, as Model 2 can *emulate* the Carreras (2007) algorithm by "demoting" each third-order part into a second-order part:

$$\textsc{ScoreGS}(\boldsymbol{x}, g, h, m, s) = \textsc{ScoreG}(\boldsymbol{x}, g, h, m)$$
$$\textsc{ScoreTS}(\boldsymbol{x}, h, m, s, t) = \textsc{ScoreS}(\boldsymbol{x}, h, m, s)$$

where ScoreG, ScoreS, ScoreGS and ScoreTS are the scoring functions for grandchildren, siblings, grand-siblings and tri-siblings, respectively. The emulated version has the same computational complexity as the original, so there is no practical reason to prefer it over the original. Nevertheless, the relationship illustrated above highlights the efficiency of our approach: we are able to recover third-order parts in place of second-order parts, at no additional cost.

4.4 Discussion

The technique of grandparent-index augmentation has proven fruitful, as it allows us to parse expressive third-order factorizations while retaining an efficient $O(n^4)$ runtime. In fact, our third-order parsing algorithms are "optimally" efficient in an asymptotic sense. Since each third-order part is composed of four separate indices, there are $\Theta(n^4)$ distinct parts. Any third-order parsing algorithm must at least consider the score of each part, hence third-order parsing is $\Omega(n^4)$ and it follows that the asymptotic complexity of Models 1 and 2 cannot be improved.

The key to the efficiency of our approach is a fundamental asymmetry in the structure of a directed tree: a head can have any number of modifiers, while a modifier always has exactly one head. Factorizations like that of Carreras (2007) obtain grandchild parts by augmenting spans with the indices of modifiers, leading to limitations on

[4] The reason for the restriction is that in Model 2, grand-siblings can only be derived via Figure 7(b), which does not recursively copy the grandparent index for reuse in smaller g-spans as Model 1 does in Figure 6(b).

the grandchildren that can participate in the factorization. Our method, by "inverting" the modifier indices into grandparent indices, exploits the structural asymmetry.

As a final note, the parsing algorithms described in this section fall into the category of *projective* dependency parsers, which forbid crossing dependencies. If crossing dependencies are allowed, it is possible to parse a first-order factorization by finding the maximum directed spanning tree (Chu and Liu, 1965; Edmonds, 1967; McDonald et al., 2005b). Unfortunately, designing efficient higher-order non-projective parsers is likely to be challenging, based on recent hardness results (McDonald and Pereira, 2006; McDonald and Satta, 2007).

5 Extensions

We briefly outline a few extensions to our algorithms; we hope to explore these in future work.

5.1 Probabilistic inference

Many statistical modeling techniques are based on partition functions and marginals—summations over the set of possible trees $\mathcal{Y}(\boldsymbol{x})$. Straightforward adaptations of the inside-outside algorithm (Baker, 1979) to our dynamic-programming structures would suffice to compute these quantities.

5.2 Labeled parsing

Our parsers are easily extended to labeled dependencies. Direct integration of labels into Models 1 and 2 would result in third-order parts composed of three labeled dependencies, at the cost of increasing the time and space complexities by factors of $O(L^3)$ and $O(L^2)$, respectively, where L bounds the number of labels per dependency.

5.3 Word senses

If each word in $\boldsymbol{x}$ has a set of possible "senses," our parsers can be modified to recover the best joint assignment of syntax and senses for $\boldsymbol{x}$, by adapting methods in Eisner (2000). Complexity would increase by factors of $O(S^4)$ time and $O(S^3)$ space, where S bounds the number of senses per word.

5.4 Increased context

If more vertical context is desired, the dynamic-programming structures can be extended with additional ancestor indices, resulting in a "spine" of ancestors above each span. Each additional ancestor lengthens the vertical scope of the factorization (e.g., from grand-siblings to "great-grand-siblings"), while increasing complexity by a factor of $O(n)$. Horizontal context can also be increased by adding internal sibling indices; each additional sibling widens the scope of the factorization (e.g., from grand-siblings to "grand-tri-siblings"), while increasing complexity by a factor of $O(n)$.

6 Related work

Our method augments each span with the index of the head that governs that span, in a manner superficially similar to parent annotation in CFGs (Johnson, 1998). However, parent annotation is a *grammar* transformation that is independent of any particular sentence, whereas our method annotates spans with *indices* into the current sentence. These indices allow the use of arbitrary features predicated on the position of the grandparent (e.g., word identity, POS tag, contextual POS tags) without affecting the asymptotic complexity of the parsing algorithm. Efficiently encoding this kind of information into a sentence-independent grammar transformation would be challenging at best.

Eisner (2000) defines dependency parsing models where each word has a set of possible "senses" and the parser recovers the best joint assignment of syntax and senses. Our new parsing algorithms could be implemented by defining the "sense" of each word as the index of its head. However, when parsing with senses, the complexity of the Eisner (2000) parser increases by factors of $O(S^3)$ time and $O(S^2)$ space (ibid., Section 4.2). Since each word has n potential heads, a direct application of the word-sense parser leads to time and space complexities of $O(n^6)$ and $O(n^4)$, respectively, in contrast to our $O(n^4)$ and $O(n^3)$.[5]

Eisner (2000) also uses head automata to score or recognize the dependents of each head. An interesting question is whether these automata could be coerced into modeling the grandparent indices used in our parsing algorithms. However, note that the head automata are defined in a sentence-independent manner, with two automata per word in the vocabulary (ibid., Section 2). The automata are thus analogous to the rules of a CFG and at-

[5] In brief, the reason for the inefficiency is that the word-sense parser is unable to exploit certain constraints, such as the fact that the endpoints of a sibling g-span must have the same head. The word-sense parser would needlessly enumerate all possible pairs of heads in this case.

tempts to use them to model grandparent indices would face difficulties similar to those already described for grammar transformations in CFGs.

It should be noted that third-order parsers have previously been proposed by McDonald and Pereira (2006), who remarked that their second-order sibling parser (see Figure 3) could easily be extended to capture $m > 1$ successive modifiers in $O(n^{m+1})$ time (ibid., Section 2.2). To our knowledge, however, Models 1 and 2 are the first third-order parsing algorithms capable of modeling grandchild parts. In our experiments, we find that grandchild interactions make important contributions to parsing performance (see Table 3).

Carreras (2007) presents a second-order parser that can score both sibling and grandchild parts, with complexities of $O(n^4)$ time and $O(n^3)$ space. An important limitation of the parser's factorization is that it only defines grandchild parts for outermost grandchildren: (g, h, m) is scored only when m is the outermost modifier of h in some direction. Note that Models 1 and 2 have the same complexity as Carreras (2007), but strictly greater expressiveness: for each sibling or grandchild part used in the Carreras (2007) factorization, Model 1 defines an enclosing grand-sibling, while Model 2 defines an enclosing tri-sibling or grand-sibling.

The factored parsing approach we focus on is sometimes referred to as "graph-based" parsing; a popular alternative is "transition-based" parsing, in which trees are constructed by making a series of incremental decisions (Yamada and Matsumoto, 2003; Attardi, 2006; Nivre et al., 2006; McDonald and Nivre, 2007). Transition-based parsers do not impose factorizations, so they can define arbitrary features on the tree as it is being built. As a result, however, they rely on greedy or approximate search algorithms to solve Eq. 1.

7 Parsing experiments

In order to evaluate the effectiveness of our parsers in practice, we apply them to the Penn WSJ Treebank (Marcus et al., 1993) and the Prague Dependency Treebank (Hajič et al., 2001; Hajič, 1998).[6] We use standard training, validation, and test splits[7] to facilitate comparisons. Accuracy is measured with unlabeled attachment score (UAS): the percentage of words with the correct head.[8]

7.1 Features for third-order parsing

Our parsing algorithms can be applied to scores originating from any source, but in our experiments we chose to use the framework of structured linear models, deriving our scores as:

$$\textsc{ScorePart}(\boldsymbol{x}, p) \;=\; \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, p)$$

Here, $\boldsymbol{f}$ is a feature-vector mapping and $\boldsymbol{w}$ is a vector of associated parameters. Following standard practice for higher-order dependency parsing (McDonald and Pereira, 2006; Carreras, 2007), Models 1 and 2 evaluate not only the relevant third-order parts, but also the lower-order parts that are implicit in their third-order factorizations. For example, Model 1 defines feature mappings for dependencies, siblings, grandchildren, and grand-siblings, so that the score of a dependency parse is given by:

$$\begin{aligned}\textsc{Model1Score}(\boldsymbol{x}, y) \;=\; & \\ & \sum_{(h,m)\in y} \boldsymbol{w}_{\text{dep}} \cdot \boldsymbol{f}_{\text{dep}}(\boldsymbol{x}, h, m) \\ & \sum_{(h,m,s)\in y} \boldsymbol{w}_{\text{sib}} \cdot \boldsymbol{f}_{\text{sib}}(\boldsymbol{x}, h, m, s) \\ & \sum_{(g,h,m)\in y} \boldsymbol{w}_{\text{gch}} \cdot \boldsymbol{f}_{\text{gch}}(\boldsymbol{x}, g, h, m) \\ & \sum_{(g,h,m,s)\in y} \boldsymbol{w}_{\text{gsib}} \cdot \boldsymbol{f}_{\text{gsib}}(\boldsymbol{x}, g, h, m, s)\end{aligned}$$

Above, y is simultaneously decomposed into several different types of parts; trivial modifications to the Model 1 parser allow it to evaluate all of the necessary parts in an interleaved fashion. A similar treatment of Model 2 yields five feature mappings: the four above plus $\boldsymbol{f}_{\text{tsib}}(\boldsymbol{x}, h, m, s, t)$, which represents tri-sibling parts.

The lower-order feature mappings $\boldsymbol{f}_{\text{dep}}$, $\boldsymbol{f}_{\text{sib}}$, and $\boldsymbol{f}_{\text{gch}}$ are based on feature sets from previous work (McDonald et al., 2005a; McDonald and Pereira, 2006; Carreras, 2007), to which we added lexicalized versions of several features. For example, $\boldsymbol{f}_{\text{dep}}$ contains lexicalized "in-between" features that depend on the head and modifier words as well as a word lying in between the two; in contrast, previous work has generally defined in-between features for POS tags only. As another example, our

[6] For English, we extracted dependencies using Joakim Nivre's Penn2Malt tool with standard head rules (Yamada and Matsumoto, 2003); for Czech, we "projectivized" the training data by finding best-match projective trees.

[7] For Czech, the PDT has a predefined split; for English, we split the Sections as: 2–21 training, 22 validation, 23 test.

[8] As in previous work, English evaluation ignores any token whose gold-standard POS tag is one of {`` '' : , .}.

second-order mappings $\boldsymbol{f}_{\text{sib}}$ and $\boldsymbol{f}_{\text{gch}}$ define lexical trigram features, while previous work has generally used POS trigrams only.

Our third-order feature mappings $\boldsymbol{f}_{\text{gsib}}$ and $\boldsymbol{f}_{\text{tsib}}$ consist of four types of features. First, we define *4-gram features* that characterize the four relevant indices using words and POS tags; examples include POS 4-grams and mixed 4-grams with one word and three POS tags. Second, we define *4-gram context features* consisting of POS 4-grams augmented with adjacent POS tags: for example, $\boldsymbol{f}_{\text{gsib}}(\boldsymbol{x}, g, h, m, s)$ includes POS 7-grams for the tags at positions $(g, h, m, s, g{+}1, h{+}1, m{+}1)$. Third, we define *backed-off features* that track bigram and trigram interactions which are absent in the lower-order feature mappings: for example, $\boldsymbol{f}_{\text{tsib}}(\boldsymbol{x}, h, m, s, t)$ contains features predicated on the trigram (m, s, t) and the bigram (m, t), neither of which exist in any lower-order part. Fourth, noting that coordinations are typically annotated as grand-siblings (e.g., "report purchases and sales" in Figure 1), we define *coordination features* for certain grand-sibling parts. For example, $\boldsymbol{f}_{\text{gsib}}(\boldsymbol{x}, g, h, m, s)$ contains features examining the implicit head-modifier relationship (g, m) that are only activated when the POS tag of s is a coordinating conjunction.

Finally, we make two brief remarks regarding the use of POS tags. First, we assume that input sentences have been automatically tagged in a preprocessing step.[9] Second, for any feature that depends on POS tags, we include *two* copies of the feature: one using normal POS tags and another using coarsened versions[10] of the POS tags.

7.2 Averaged perceptron training

There are a wide variety of parameter estimation methods for structured linear models, such as log-linear models (Lafferty et al., 2001) and max-margin models (Taskar et al., 2003). We chose the averaged structured perceptron (Freund and Schapire, 1999; Collins, 2002) as it combines highly competitive performance with fast training times, typically converging in 5–10 iterations. We train each parser for 10 iterations and select parameters from the iteration that achieves the best score on the validation set.

Beam	Pass	Orac	Acc1	Acc2	Time1	Time2
0.0001	26.5	99.92	93.49	93.49	49.6m	73.5m
0.001	16.7	99.72	93.37	93.29	25.9m	24.2m
0.01	9.1	99.19	93.26	93.16	6.7m	7.9m

Table 1: Effect of the marginal-probability beam on English parsing. For each beam value, parsers were trained on the English training set and evaluated on the English validation set; the same beam value was applied to both training and validation data. **Pass** = %dependencies surviving the beam in training data, **Orac** = maximum achievable UAS on validation data, **Acc1/Acc2** = UAS of Models 1/2 on validation data, and **Time1/Time2** = minutes per perceptron training iteration for Models 1/2, averaged over all 10 iterations. For perspective, the English training set has a total of 39,832 sentences and 950,028 words. A beam of 0.0001 was used in all experiments outside this table.

7.3 Coarse-to-fine pruning

In order to decrease training times, we follow Carreras et al. (2008) and eliminate unlikely dependencies using a form of coarse-to-fine pruning (Charniak and Johnson, 2005; Petrov and Klein, 2007). In brief, we train a log-linear first-order parser[11] and for every sentence $\boldsymbol{x}$ in training, validation, and test data we compute the marginal probability $P(h, m \mid \boldsymbol{x})$ of each dependency. Our parsers are then modified to ignore any dependency (h, m) whose marginal probability is below $0.0001 \times \max_{h'} P(h', m \mid \boldsymbol{x})$. Table 1 provides information on the behavior of the pruning method.

7.4 Main results

Table 2 lists the accuracy of Models 1 and 2 on the English and Czech test sets, together with some relevant results from related work.[12] The models marked "†" are *not* directly comparable to our work as they depend on additional sources of information that our models are trained without—unlabeled data in the case of Koo et al. (2008) and

[9] For Czech, the PDT provides automatic tags; for English, we used MXPOST (Ratnaparkhi, 1996) to tag validation and test data, with 10-fold cross-validation on the training set. Note that the reliance on POS-tagged input can be relaxed slightly by treating POS tags as word senses; see Section 5.3 and McDonald (2006, Table 6.1).

[10] For Czech, we used the first character of the tag; for English, we used the first two characters, except `PRP` and `PRP$`.

[11] For English, we generate marginals using a projective parser (Baker, 1979; Eisner, 2000); for Czech, we generate marginals using a non-projective parser (Smith and Smith, 2007; McDonald and Satta, 2007; Koo et al., 2007). Parameters for these models are obtained by running exponentiated gradient training for 10 iterations (Collins et al., 2008).

[12] Model 0 was not tested as its factorization is a strict subset of the factorization of Model 1.

Parser	Eng	Cze
McDonald et al. (2005a,2005b)	90.9	84.4
McDonald and Pereira (2006)	91.5	85.2
Koo et al. (2008), standard	92.02	86.13
Model 1	93.04	87.38
Model 2	92.93	87.37
Koo et al. (2008), semi-sup†	93.16	87.13
Suzuki et al. (2009)†	93.79	88.05
Carreras et al. (2008)†	93.5	—

Table 2: UAS of Models 1 and 2 on test data, with relevant results from related work. Note that Koo et al. (2008) is listed with standard features and semi-supervised features. †: see main text.

Suzuki et al. (2009) and phrase-structure annotations in the case of Carreras et al. (2008). All three of the "†" models are based on versions of the Carreras (2007) parser, so modifying these methods to work with our new third-order parsing algorithms would be an interesting topic for future research. For example, Models 1 and 2 obtain results comparable to the semi-supervised parsers of Koo et al. (2008), and additive gains might be realized by applying their cluster-based feature sets to our enriched factorizations.

7.5 Ablation studies

In order to better understand the contributions of the various feature types, we ran additional ablation experiments; the results are listed in Table 3, in addition to the scores of Model 0 and the emulated Carreras (2007) parser (see Section 4.3). Interestingly, grandchild interactions appear to provide important information: for example, when Model 2 is used without grandchild-based features ("Model 2, no-G" in Table 3), its accuracy suffers noticeably. In addition, it seems that grandchild interactions are particularly useful in Czech, while sibling interactions are less important: consider that Model 0, a second-order grandchild parser with no sibling-based features, can easily outperform "Model 2, no-G," a *third*-order sibling parser with no grandchild-based features.

8 Conclusion

We have presented new parsing algorithms that are capable of efficiently parsing third-order factorizations, including both grandchild and sibling interactions. Due to space restrictions, we have been necessarily brief at some points in this paper; some additional details can be found in Koo (2010).

Parser	Eng	Cze
Model 0	93.07	87.39
Carreras (2007) emulation	93.14	87.25
Model 1	93.49	87.64
Model 1, no-3rd	93.17	87.57
Model 2	93.49	87.46
Model 2, no-3rd	93.20	87.43
Model 2, no-G	92.92	86.76

Table 3: UAS for modified versions of our parsers on validation data. The term **no-3rd** indicates a parser that was trained and tested with the third-order feature mappings $\boldsymbol{f}_{\text{gsib}}$ and $\boldsymbol{f}_{\text{tsib}}$ deactivated, though lower-order features were retained; note that "Model 2, no-3rd" is not identical to the Carreras (2007) parser as it defines grandchild parts for the pair of grandchildren. The term **no-G** indicates a parser that was trained and tested with the grandchild-based feature mappings $\boldsymbol{f}_{\text{gch}}$ and $\boldsymbol{f}_{\text{gsib}}$ deactivated; note that "Model 2, no-G" emulates the third-order sibling parser proposed by McDonald and Pereira (2006).

There are several possibilities for further research involving our third-order parsing algorithms. One idea would be to consider extensions and modifications of our parsers, some of which have been suggested in Sections 5 and 7.4. A second area for future work lies in applications of dependency parsing. While we have evaluated our new algorithms on standard parsing benchmarks, there are a wide variety of tasks that may benefit from the extended context offered by our third-order factorizations; for example, the 4-gram substructures enabled by our approach may be useful for dependency-based language modeling in machine translation (Shen et al., 2008). Finally, in the hopes that others in the NLP community may find our parsers useful, we provide a free distribution of our implementation.[2]

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments and suggestions. We also thank Regina Barzilay and Alexander Rush for their much-appreciated input during the writing process. The authors gratefully acknowledge the following sources of support: Terry Koo and Michael Collins were both funded by a DARPA subcontract under SRI (#27-001343), and Michael Collins was additionally supported by NTT (Agmt. dtd. 06/21/98).

References

Giuseppe Attardi. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of the 10th CoNLL*, pages 166–170. Association for Computational Linguistics.

James Baker. 1979. Trainable Grammars for Speech Recognition. In *Proceedings of the 97th meeting of the Acoustical Society of America*.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing. In *Proceedings of the 12th CoNLL*, pages 9–16. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd ACL*.

Y.J. Chu and T.H. Liu. 1965. On the Shortest Arborescence of a Directed Graph. *Science Sinica*, 14:1396–1400.

John Cocke and Jacob T. Schwartz. 1970. Programming Languages and Their Compilers: Preliminary Notes. Technical report, New York University.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. 2008. Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks. *Journal of Machine Learning Research*, 9:1775–1822, Aug.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 7th EMNLP*, pages 1–8. Association for Computational Linguistics.

Jack R. Edmonds. 1967. Optimum Branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th COLING*, pages 340–345. Association for Computational Linguistics.

Jason Eisner. 2000. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Yoav Freund and Robert E. Schapire. 1999. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevova, and Petr Sgall. 2001. *The Prague Dependency Treebank 1.0, LDC No. LDC2001T10*. Linguistics Data Consortium.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19.

Mark Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632.

Tadao Kasami. 1965. An Efficient Recognition and Syntax-analysis Algorithm for Context-free Languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured Prediction Models via the Matrix-Tree Theorem. In *Proceedings of EMNLP-CoNLL*, pages 141–150. Association for Computational Linguistics.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of the 46th ACL*, pages 595–603. Association for Computational Linguistics.

Terry Koo. 2010. *Advances in Discriminative Dependency Parsing*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th ICML*, pages 282–289. Morgan Kaufmann.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David A. McAllester. 1999. On the Complexity Analysis of Static Analyses. In *Proceedings of the 6th Static Analysis Symposium*, pages 312–329. Springer-Verlag.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsers. In *Proceedings of EMNLP-CoNLL*, pages 122–131. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of the 11th EACL*, pages 81–88. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the Complexity of Non-Projective Data-Driven Dependency Parsing. In *Proceedings of IWPT*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd ACL*, pages 91–98. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530. Association for Computational Linguistics.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA, July.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of the 10th CoNLL*, pages 221–225. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of HLT-NAACL*, pages 404–411. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the 1st EMNLP*, pages 133–142. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the 46th ACL*, pages 577–585. Association for Computational Linguistics.

David A. Smith and Noah A. Smith. 2007. Probabilistic Models of Nonprojective Dependency Trees. In *Proceedings of EMNLP-CoNLL*, pages 132–140. Association for Computational Linguistics.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In *Proceedings of EMNLP*, pages 551–560. Association for Computational Linguistics.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max margin markov networks. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the 8th IWPT*, pages 195–206. Association for Computational Linguistics.

David H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Dependency Parsing and Projection Based on Word-Pair Classification

Wenbin Jiang and **Qun Liu**

Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{jiangwenbin, liuqun}@ict.ac.cn

Abstract

In this paper we describe an intuitionistic method for dependency parsing, where a classifier is used to determine whether a pair of words forms a dependency edge. And we also propose an effective strategy for dependency projection, where the dependency relationships of the word pairs in the source language are projected to the word pairs of the target language, leading to a set of classification instances rather than a complete tree. Experiments show that, the classifier trained on the projected classification instances significantly outperforms previous projected dependency parsers. More importantly, when this classifier is integrated into a maximum spanning tree (MST) dependency parser, obvious improvement is obtained over the MST baseline.

1 Introduction

Supervised dependency parsing achieves the state-of-the-art in recent years (McDonald et al., 2005a; McDonald and Pereira, 2006; Nivre et al., 2006). Since it is costly and difficult to build human-annotated treebanks, a lot of works have also been devoted to the utilization of unannotated text. For example, the unsupervised dependency parsing (Klein and Manning, 2004) which is totally based on unannotated data, and the semisupervised dependency parsing (Koo et al., 2008) which is based on both annotated and unannotated data. Considering the higher complexity and lower performance in unsupervised parsing, and the need of reliable priori knowledge in semisupervised parsing, it is a promising strategy to project the dependency structures from a resource-rich language to a resource-scarce one across a bilingual corpus (Hwa et al., 2002; Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009; Jiang et al., 2009).

For dependency projection, the relationship between words in the parsed sentences can be simply projected across the word alignment to words in the unparsed sentences, according to the DCA assumption (Hwa et al., 2005). Such a projection procedure suffers much from the word alignment errors and syntactic isomerism between languages, which usually lead to relationship projection conflict and incomplete projected dependency structures. To tackle this problem, Hwa et al. (2005) use some filtering rules to reduce noise, and some hand-designed rules to handle language heterogeneity. Smith and Eisner (2009) perform dependency projection and annotation adaptation with quasi-synchronous grammar features. Jiang and Liu (2009) resort to a dynamic programming procedure to search for a completed projected tree. However, these strategies are all confined to the same category that dependency projection must produce completed projected trees. Because of the free translation, the syntactic isomerism between languages and word alignment errors, it would be strained to completely project the dependency structure from one language to another.

We propose an effective method for dependency projection, which does not have to produce complete projected trees. Given a word-aligned bilingual corpus with source language sentences parsed, the dependency relationships of the word pairs in the source language are projected to the word pairs of the target language. A dependency relationship is a boolean value that represents whether this word pair forms a dependency edge. Thus a set of classification instances are obtained. Meanwhile, we propose an intuitionistic model for dependency parsing, which uses a classifier to determine whether a pair of words form a dependency edge. The classifier can then be trained on the projected classification instance set, so as to build a projected dependency parser without the need of complete projected trees.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 12–20,
Uppsala, Sweden, 11-16 July 2010.

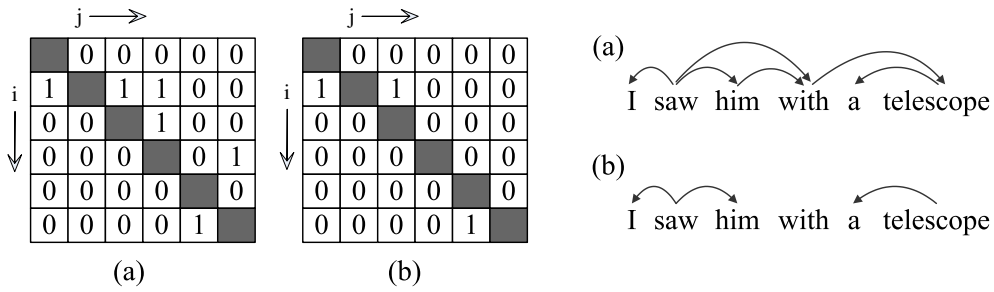


Figure 1: Illegal (a) and incomplete (b) dependency tree produced by the simple-collection method.

Experimental results show that, the classifier trained on the projected classification instances significantly outperforms the projected dependency parsers in previous works. The classifier trained on the Chinese projected classification instances achieves a precision of 58.59% on the CTB standard test set. More importantly, when this classifier is integrated into a 2nd-ordered maximum spanning tree (MST) dependency parser (McDonald and Pereira, 2006) in a weighted average manner, significant improvement is obtained over the MST baselines. For the 2nd-order MST parser trained on Penn Chinese Treebank (CTB) 5.0, the classifier give an precision increment of 0.5 points. Especially for the parser trained on the smaller CTB 1.0, more than 1 points precision increment is obtained.

In the rest of this paper, we first describe the word-pair classification model for dependency parsing (section 2) and the generation method of projected classification instances (section 3). Then we describe an application of the projected parser: boosting a state-of-the-art 2nd-ordered MST parser (section 4). After the comparisons with previous works on dependency parsing and projection, we finally five the experimental results.

2 Word-Pair Classification Model

2.1 Model Definition

Following (McDonald et al., 2005a), $\mathbf{x}$ is used to denote the sentence to be parsed, and x_i to denote the i-th word in the sentence. $\mathbf{y}$ denotes the dependency tree for sentence $\mathbf{x}$, and $(i,j) \in \mathbf{y}$ represents a dependency edge from word x_i to word x_j, where x_i is the parent of x_j.

The task of the word-pair classification model is to determine whether any candidate word pair, x_i and x_j s.t. $1 \leq i,j \leq |\mathbf{x}|$ and $i \neq j$, forms a dependency edge. The classification result $\mathcal{C}(i,j)$ can be a boolean value:

$$\mathcal{C}(i,j) = p \qquad p \in \{0,1\} \tag{1}$$

as produced by a support vector machine (SVM) classifier (Vapnik, 1998). $p = 1$ indicates that the classifier supports the candidate edge (i,j), and $p = 0$ the contrary. $\mathcal{C}(i,j)$ can also be a real-valued probability:

$$\mathcal{C}(i,j) = p \qquad 0 \leq p \leq 1 \tag{2}$$

as produced by an maximum entropy (ME) classifier (Berger et al., 1996). p is a probability which indicates the degree the classifier support the candidate edge (i,j). Ideally, given the classification results for all candidate word pairs, the dependency parse tree can be composed of the candidate edges with higher score (1 for the boolean-valued classifier, and large p for the real-valued classifier). However, more robust strategies should be investigated since the ambiguity of the language syntax and the classification errors usually lead to illegal or incomplete parsing result, as shown in Figure 1.

Follow the edge based factorization method (Eisner, 1996), we factorize the score of a dependency tree $\mathbf{s}(\mathbf{x},\mathbf{y})$ into its dependency edges, and design a dynamic programming algorithm to search for the candidate parse with maximum score. This strategy alleviate the classification errors to some degree and ensure a valid, complete dependency parsing tree. If a boolean-valued classifier is used, the search algorithm can be formalized as:

$$\begin{aligned} \tilde{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \mathbf{s}(\mathbf{x},\mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \sum_{(i,j)\in\mathbf{y}} \mathcal{C}(i,j) \end{aligned} \tag{3}$$

And if a probability-valued classifier is used instead, we replace the accumulation with cumula-

Type	Features		
Unigram	$word_i \circ pos_i$	$word_i$	pos_i
	$word_j \circ pos_j$	$word_j$	pos_j
Bigram	$word_i \circ pos_i \circ word_j \circ pos_j$	$pos_i \circ word_j \circ pos_j$	$word_i \circ word_j \circ pos_j$
	$word_i \circ pos_i \circ pos_j$	$word_i \circ pos_i \circ word_j$	$word_i \circ word_j$
	$pos_i \circ pos_j$	$word_i \circ pos_j$	$pos_i \circ word_j$
Surrounding	$pos_i \circ pos_{i+1} \circ pos_{j-1} \circ pos_j$	$pos_{i-1} \circ pos_i \circ pos_{j-1} \circ pos_j$	$pos_i \circ pos_{i+1} \circ pos_j \circ pos_{j+1}$
	$pos_{i-1} \circ pos_i \circ pos_j \circ pos_{j+1}$	$pos_{i-1} \circ pos_i \circ pos_{j-1}$	$pos_{i-1} \circ pos_i \circ pos_{j+1}$
	$pos_i \circ pos_{i+1} \circ pos_{j-1}$	$pos_i \circ pos_{i+1} \circ pos_{j+1}$	$pos_{i-1} \circ pos_{j-1} \circ pos_j$
	$pos_{i-1} \circ pos_j \circ pos_{j+1}$	$pos_{i+1} \circ pos_{j-1} \circ pos_j$	$pos_{i+1} \circ pos_j \circ pos_{j+1}$
	$pos_i \circ pos_{j-1} \circ pos_j$	$pos_i \circ pos_j \circ pos_{j+1}$	$pos_{i-1} \circ pos_i \circ pos_j$
	$pos_i \circ pos_{i+1} \circ pos_j$		

Table 1: Feature templates for the word-pair classification model.

tive product:

$$\begin{aligned} \tilde{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \mathbf{s}(\mathbf{x}, \mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \prod_{(i,j)\in\mathbf{y}} \mathcal{C}(i,j) \end{aligned} \tag{4}$$

Where y is searched from the set of well-formed dependency trees.

In our work we choose a real-valued ME classifier. Here we give the calculation of dependency probability $\mathcal{C}(i,j)$. We use $\mathbf{w}$ to denote the parameter vector of the ME model, and $\mathbf{f}(i,j,r)$ to denote the feature vector for the *assumption* that the word pair i and j has a dependency relationship r. The symbol r indicates the supposed classification result, where $r = +$ means we suppose it as a dependency edge and $r = -$ means the contrary. A feature $\mathbf{f}_k(i,j,r) \in \mathbf{f}(i,j,r)$ equals 1 if it is activated by the assumption and equals 0 otherwise. The dependency probability can then be defined as:

$$\begin{aligned} \mathcal{C}(i,j) &= \frac{exp(\mathbf{w}\cdot\mathbf{f}(i,j,+))}{\sum_r exp(\mathbf{w}\cdot\mathbf{f}(i,j,r))} \\ &= \frac{exp(\sum_k \mathbf{w}_k \times \mathbf{f}_k(i,j,+))}{\sum_r exp(\sum_k \mathbf{w}_k \times \mathbf{f}_k(i,j,r))} \end{aligned} \tag{5}$$

2.2 Features for Classification

The feature templates for the classifier are similar to those of 1st-ordered MST model (McDonald et al., 2005a). [1] Each feature is composed of some words and POS tags surrounded word i and/or word j, as well as an optional distance representations between this two words. Table shows the feature templates we use.

Previous graph-based dependency models usually use the index distance of word i and word j to enrich the features with word distance information. However, in order to utilize some syntax information between the pair of words, we adopt the syntactic distance representation of (Collins, 1996), named *Collins distance* for convenience. A Collins distance comprises the answers of 6 questions:

- Does word i precede or follow word j?
- Are word i and word j adjacent?
- Is there a verb between word i and word j?
- Are there 0, 1, 2 or more than 2 commas between word i and word j?
- Is there a comma immediately following the first of word i and word j?
- Is there a comma immediately preceding the second of word i and word j?

Besides the original features generated according to the templates in Table 1, the enhanced features with Collins distance as postfixes are also used in training and decoding of the word-pair classifier.

2.3 Parsing Algorithm

We adopt logarithmic dependency probabilities in decoding, therefore the cumulative product of probabilities in formula 6 can be replaced by accumulation of logarithmic probabilities:

$$\begin{aligned} \tilde{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} \mathbf{s}(\mathbf{x}, \mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \prod_{(i,j)\in\mathbf{y}} \mathcal{C}(i,j) \\ &= \operatorname*{argmax}_{\mathbf{y}} \sum_{(i,j)\in\mathbf{y}} log(\mathcal{C}(i,j)) \end{aligned} \tag{6}$$

Thus, the decoding algorithm for 1st-ordered MST model, such as the Chu-Liu-Edmonds algorithm

[1] We exclude the *in between* features of McDonald et al. (2005a) since preliminary experiments show that these features bring no improvement to the word-pair classification model.

Algorithm 1 Dependency Parsing Algorithm.

1: **Input**: sentence $\mathbf{x}$ to be parsed
2: **for** $\langle i, j \rangle \subseteq \langle 1, |\mathbf{x}| \rangle$ in topological order **do**
3: $\quad \mathbf{buf} \leftarrow \emptyset$
4: $\quad$ **for** $k \leftarrow i..j-1$ **do** ▷ all partitions
5: $\quad\quad$ **for** $l \in \mathbf{V}[i,k]$ and $r \in \mathbf{V}[k+1,j]$ **do**
6: $\quad\quad\quad$ insert DERIV(l, r) into $\mathbf{buf}$
7: $\quad\quad\quad$ insert DERIV(r, l) into $\mathbf{buf}$
8: $\quad \mathbf{V}[i,j] \leftarrow$ top K derivations of $\mathbf{buf}$
9: **Output:** the best derivation of $\mathbf{V}[1, |\mathbf{x}|]$
10: **function** DERIV(p, c)
11: $\quad d \leftarrow p \cup c \cup \{(p \cdot root, c \cdot root)\}$ ▷ new derivation
12: $\quad d \cdot evl \leftarrow$ EVAL(d) ▷ evaluation function
13: $\quad$ **return** d

used in McDonald et al. (2005b), is also applicable here. In this work, however, we still adopt the more general, bottom-up dynamic programming algorithm Algorithm 1 in order to facilitate the possible expansions. Here, $\mathbf{V}[i,j]$ contains the candidate parsing segments of the span $[i,j]$, and the function EVAL(d) accumulates the scores of all the edges in dependency segment d. In practice, the cube-pruning strategy (Huang and Chiang, 2005) is used to speed up the enumeration of derivations (loops started by line 4 and 5).

3 Projected Classification Instance

After the introduction of the word-pair classification model, we now describe the extraction of projected dependency instances. In order to alleviate the effect of word alignment errors, we base the projection on the alignment matrix, a compact representation of multiple GIZA++ (Och and Ney, 2000) results, rather than a single word alignment in previous dependency projection works. Figure 2 shows an example.

Suppose a bilingual sentence pair, composed of a source sentence $\mathbf{e}$ and its target translation $\mathbf{f}$. $\mathbf{y_e}$ is the parse tree of the source sentence. $\mathbf{A}$ is the alignment matrix between them, and each element $\mathbf{A}_{i,j}$ denotes the degree of the alignment between word $\mathbf{e}_i$ and word $\mathbf{f}_j$. We define a boolean-valued function $\delta(\mathbf{y}, i, j, r)$ to investigate the dependency relationship of word i and word j in parse tree $\mathbf{y}$:

$$\delta(\mathbf{y},i,j,r) = \begin{cases} 1 & (i,j) \in \mathbf{y} \textbf{ and } r = + \\ & \quad \textbf{or} \\ & (i,j) \notin \mathbf{y} \textbf{ and } r = - \\ 0 & \textbf{otherwise} \end{cases} \quad (7)$$

Then the score that word i and word j in the target sentence $\mathbf{y}$ forms a projected dependency edge, $\mathbf{s}_+(i,j)$, can be defined as:

$$\mathbf{s}_+(i,j) = \sum_{i',j'} \mathbf{A}_{i,i'} \times \mathbf{A}_{j,j'} \times \delta(\mathbf{y_e}, i', j', +) \quad (8)$$

The score that they do not form a projected dependency edge can be defined similarly:

$$\mathbf{s}_-(i,j) = \sum_{i',j'} \mathbf{A}_{i,i'} \times \mathbf{A}_{j,j'} \times \delta(\mathbf{y_e}, i', j', -) \quad (9)$$

Note that for simplicity, the condition factors $\mathbf{y_e}$ and $\mathbf{A}$ are omitted from these two formulas. We finally define the probability of the supposed projected dependency edge as:

$$\mathcal{C}_p(i,j) = \frac{exp(\mathbf{s}_+(i,j))}{exp(\mathbf{s}_+(i,j)) + exp(\mathbf{s}_-(i,j))} \quad (10)$$

The probability $\mathcal{C}_p(i,j)$ is a real value between 0 and 1. Obviously, $\mathcal{C}_p(i,j) = 0.5$ indicates the most ambiguous case, where we can not distinguish between positive and negative at all. On the other hand, there are as many as $2|\mathbf{f}|(|\mathbf{f}|-1)$ candidate projected dependency instances for the target sentence $\mathbf{f}$. Therefore, we need choose a threshold b for $\mathcal{C}_p(i,j)$ to filter out the ambiguous instances: the instances with $\mathcal{C}_p(i,j) > b$ are selected as the positive, and the instances with $\mathcal{C}_p(i,j) < 1-b$ are selected as the negative.

	1	saw	him	with	a	telescope
wo	0.90	0.00	0.15	0.00	0.05	0.00
yong	0.05	0.00	0.00	0.95	0.05	0.00
wangyuanjing	0.00	0.10	0.00	0.05	0.10	0.95
kanjian	0.00	0.85	0.00	0.15	0.10	0.00
le	0.00	0.30	0.00	0.00	0.00	0.05
ta	0.05	0.00	0.95	0.00	0.00	0.00

Figure 2: The word alignment matrix between a Chinese sentence and its English translation. Note that probabilities need not to be normalized across rows or columns.

4 Boosting an MST Parser

The classifier can be used to boost a existing parser trained on human-annotated trees. We first establish a unified framework for the enhanced parser. For a sentence to be parsed, $\mathbf{x}$, the enhanced parser selects the best parse $\tilde{\mathbf{y}}$ according to both the baseline model $\mathbb{B}$ and the projected classifier $\mathbb{C}$.

$$\tilde{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}}[\mathbf{s}_{\mathbb{B}}(\mathbf{x},\mathbf{y}) + \lambda \mathbf{s}_{\mathbb{C}}(\mathbf{x},\mathbf{y})] \quad (11)$$

Here, $\mathbf{s}_{\mathbb{B}}$ and $\mathbf{s}_{\mathbb{C}}$ denote the evaluation functions of the baseline model and the projected classifier, respectively. The parameter λ is the relative weight of the projected classifier against the baseline model.

There are several strategies to integrate the two evaluation functions. For example, they can be integrated deeply at each decoding step (Carreras et al., 2008; Zhang and Clark, 2008; Huang, 2008), or can be integrated shallowly in a reranking manner (Collins, 2000; Charniak and Johnson, 2005). As described previously, the score of a dependency tree given by a word-pair classifier can be factored into each candidate dependency edge in this tree. Therefore, the projected classifier can be integrated with a baseline model deeply at each dependency edge, if the evaluation score given by the baseline model can also be factored into dependency edges.

We choose the 2nd-ordered MST model (McDonald and Pereira, 2006) as the baseline. Especially, the effect of the Collins distance in the baseline model is also investigated. The relative weight λ is adjusted to maximize the performance on the development set, using an algorithm similar to minimum error-rate training (Och, 2003).

5 Related Works

5.1 Dependency Parsing

Both the graph-based (McDonald et al., 2005a; McDonald and Pereira, 2006; Carreras et al., 2006) and the transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2006) parsing algorithms are related to our word-pair classification model.

Similar to the graph-based method, our model is factored on dependency edges, and its decoding procedure also aims to find a maximum spanning tree in a fully connected directed graph. From this point, our model can be classified into the graph-based category. On the training method, however, our model obviously differs from other graph-based models, that we only need a set of word-pair dependency instances rather than a regular dependency treebank. Therefore, our model is more suitable for the partially bracketed or noisy training corpus.

The most apparent similarity between our model and the transition-based category is that they all need a classifier to perform classification conditioned on a certain configuration. However, they differ from each other in the classification results. The classifier in our model predicates a dependency probability for each pair of words, while the classifier in a transition-based model gives a possible next transition operation such as *shift* or *reduce*. Another difference lies in the factorization strategy. For our method, the evaluation score of a candidate parse is factorized into each dependency edge, while for the transition-based models, the score is factorized into each transition operation.

Thanks to the reminding of the third reviewer of our paper, we find that the pairwise classification schema has also been used in Japanese dependency parsing (Uchimoto et al., 1999; Kudo and Matsumoto, 2000). However, our work shows more advantage in feature engineering, model training and decoding algorithm.

5.2 Dependency Projection

Many works try to learn parsing knowledge from bilingual corpora. Lü et al. (2002) aims to obtain Chinese bracketing knowledge via ITG (Wu, 1997) alignment. Hwa et al. (2005) and Ganchev et al. (2009) induce dependency grammar via projection from aligned bilingual corpora, and use some thresholds to filter out noise and some hand-written rules to handle heterogeneity. Smith and Eisner (2009) perform dependency projection and annotation adaptation with Quasi-Synchronous Grammar features. Jiang and Liu (2009) refer to alignment matrix and a dynamic programming search algorithm to obtain better projected dependency trees.

All previous works for dependency projection (Hwa et al., 2005; Ganchev et al., 2009; Smith and Eisner, 2009; Jiang and Liu, 2009) need complete projected trees to train the projected parsers. Because of the free translation, the word alignment errors, and the heterogeneity between two languages, it is reluctant and less effective to project the dependency tree completely to the target language sentence. On the contrary, our dependency projection strategy prefer to extract a set of dependency instances, which coincides our model's demand for training corpus. An obvious advantage of this strategy is that, we can select an appropriate filtering threshold to obtain dependency instances of good quality.

In addition, our word-pair classification model can be integrated deeply into a state-of-the-art MST dependency model. Since both of them are

Corpus	Train	Dev	Test
WSJ (section)	2-21	22	23
CTB 5.0 (chapter)	others	301-325	271-300

Table 2: The corpus partition for WSJ and CTB 5.0.

factorized into dependency edges, the integration can be conducted at each dependency edge, by weightedly averaging their evaluation scores for this dependency edge. This strategy makes better use of the projected parser while with faster decoding, compared with the cascaded approach of Jiang and Liu (2009).

6 Experiments

In this section, we first validate the word-pair classification model by experimenting on human-annotated treebanks. Then we investigate the effectiveness of the dependency projection by evaluating the projected classifiers trained on the projected classification instances. Finally, we report the performance of the integrated dependency parser which integrates the projected classifier and the 2nd-ordered MST dependency parser. We evaluate the parsing accuracy by the precision of lexical heads, which is the percentage of the words that have found their correct parents.

6.1 Word-Pair Classification Model

We experiment on two popular treebanks, the Wall Street Journal (WSJ) portion of the Penn English Treebank (Marcus et al., 1993), and the Penn Chinese Treebank (CTB) 5.0 (Xue et al., 2005). The constituent trees in the two treebanks are transformed to dependency trees according to the head-finding rules of Yamada and Matsumoto (2003). For English, we use the automatically-assigned POS tags produced by an implementation of the POS tagger of Collins (2002). While for Chinese, we just use the gold-standard POS tags following the tradition. Each treebank is splitted into three partitions, for training, development and testing, respectively, as shown in Table 2.

For a dependency tree with n words, only $n-1$ positive dependency instances can be extracted. They account for only a small proportion of all the dependency instances. As we know, it is important to balance the proportions of the positive and the negative instances for a batched-trained classifier. We define a new parameter r to denote the ratio of the negative instances relative to the positive ones.

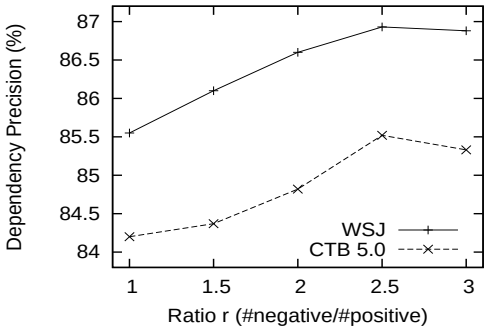


Figure 3: Performance curves of the word-pair classification model on the development sets of WSJ and CTB 5.0, with respect to a series of ratio r.

Corpus	System	P %
WSJ	Yamada and Matsumoto (2003)	90.3
	Nivre and Scholz (2004)	87.3
	1st-ordered MST	90.7
	2nd-ordered MST	91.5
	our model	86.8
CTB 5.0	1st-ordered MST	86.53
	2nd-ordered MST	87.15
	our model	82.06

Table 3: Performance of the word-pair classification model on WSJ and CTB 5.0, compared with the current state-of-the-art models.

For example, $r = 2$ means we reserve negative instances two times as many as the positive ones.

The MaxEnt toolkit by Zhang [2] is adopted to train the ME classifier on extracted instances. We set the gaussian prior as 1.0 and the iteration limit as 100, leaving other parameters as default values. We first investigate the impact of the ratio r on the performance of the classifier. Curves in Figure 3 show the performance of the English and Chinese parsers, each of which is trained on an instance set corresponding to a certain r. We find that for both English and Chinese, maximum performance is achieved at about $r = 2.5$. [3] The English and Chinese classifiers trained on the instance sets with $r = 2.5$ are used in the final evaluation phase. Table 3 shows the performances on the test sets of WSJ and CTB 5.0.

We also compare them with previous works on the same test sets. On both English and Chinese, the word-pair classification model falls behind of the state-of-the-art. We think that it is probably

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

[3] We did not investigate more fine-grained ratios, since the performance curves show no dramatic fluctuation along with the alteration of r.

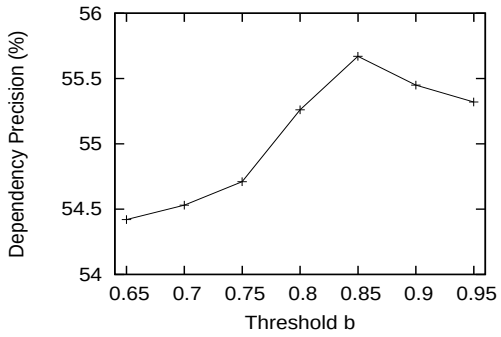


Figure 4: The performance curve of the word-pair classification model on the development set of CTB 5.0, with respect to a series of threshold b.

due to the local optimization of the training procedure. Given complete trees as training data, it is easy for previous models to utilize structural, global and linguistical information in order to obtain more powerful parameters. The main advantage of our model is that it doesn't need complete trees to tune its parameters. Therefore, if trained on instances extracted from human-annotated treebanks, the word-pair classification model would not demonstrate its advantage over existed state-of-the-art dependency parsing methods.

6.2 Dependency Projection

In this work we focus on the dependency projection from English to Chinese. We use the FBIS Chinese-English bitext as the bilingual corpus for dependency projection. It contains 239K sentence pairs with about 6.9M/8.9M words in Chinese/English. Both English and Chinese sentences are tagged by the implementations of the POS tagger of Collins (2002), which trained on WSJ and CTB 5.0 respectively. The English sentences are then parsed by an implementation of 2nd-ordered MST model of McDonald and Pereira (2006), which is trained on dependency trees extracted from WSJ. The alignment matrixes for sentence pairs are generated according to (Liu et al., 2009).

Similar to the ratio r, the threshold b need also be assigned an appropriate value to achieve a better performance. Larger thresholds result in better but less classification instances, the lower coverage of the instances would hurt the performance of the classifier. On the other hand, smaller thresholds lead to worse but more instances, and too much noisy instances will bring down the classifier's discriminating power.

We extract a series of classification instance sets with different thresholds. Then, on each instance set we train a classifier and test it on the development set of CTB 5.0. Figure 4 presents the experimental results. The curve shows that the maximum performance is achieved at the threshold of about 0.85. The classifier corresponding to this threshold is evaluated on the test set of CTB 5.0, and the test set of CTB 2.0 determined by Hwa et al. (2005). Table 4 shows the performance of the projected classifier, as well as the performance of previous works on the corresponding test sets. The projected classifier significantly outperforms previous works on both test sets, which demonstrates that the word-pair classification model, although falling behind of the state-of-the-art on human-annotated treebanks, performs well in projected dependency parsing. We give the credit to its good collaboration with the word-pair classification instance extraction for dependency projection.

Corpus	System	P %
CTB 2.0	Hwa et al. (2005)	53.9
	our model	**56.9**
CTB 5.0	Jiang and Liu (2009)	53.28
	our model	**58.59**

Table 4: The performance of the projected classifier on the test sets of CTB 2.0 and CTB 5.0, compared with the performance of previous works on the corresponding test sets.

Corpus	Baseline P%	Integrated P%
CTB 1.0	82.23	83.70
CTB 5.0	87.15	87.65

Table 5: Performance improvement brought by the projected classifier to the baseline 2nd-ordered MST parsers trained on CTB 1.0 and CTB 5.0, respectively.

6.3 Integrated Dependency Parser

We integrate the word-pair classification model into the state-of-the-art 2nd-ordered MST model. First, we implement a chart-based dynamic programming parser for the 2nd-ordered MST model, and develop a training procedure based on the perceptron algorithm with averaged parameters (Collins, 2002). On the WSJ corpus, this parser achieves the same performance as that of McDonald and Pereira (2006). Then, at each derivation step of this 2nd-ordered MST parser, we weightedly add the evaluation score given by the projected classifier to the original MST evaluation score. Such a weighted summation of two eval-

uation scores provides better evaluation for candidate parses. The weight parameter λ is tuned by a minimum error-rate training algorithm (Och, 2003).

Given a 2nd-ordered MST parser trained on CTB 5.0 as the baseline, the projected classifier brings an accuracy improvement of about 0.5 points. For the baseline trained on the smaller CTB 1.0, whose training set is chapters 1-270 of CTB 5.0, the accuracy improvement is much significant, about 1.5 points over the baseline. It indicates that, the smaller the human-annotated treebank we have, the more significant improvement we can achieve by integrating the projecting classifier. This provides a promising strategy for boosting the parsing performance of resource-scarce languages. Table 5 summarizes the experimental results.

7 Conclusion and Future Works

In this paper, we first describe an intuitionistic method for dependency parsing, which resorts to a classifier to determine whether a word pair forms a dependency edge, and then propose an effective strategy for dependency projection, which produces a set of projected classification instances rather than complete projected trees. Although this parsing method falls behind of previous models, it can collaborate well with the word-pair classification instance extraction strategy for dependency projection, and achieves the state-of-the-art in projected dependency parsing. In addition, when integrated into a 2nd-ordered MST parser, the projected parser brings significant improvement to the baseline, especially for the baseline trained on smaller treebanks. This provides a new strategy for resource-scarce languages to train high-precision dependency parsers. However, considering its lower performance on human-annotated treebanks, the dependency parsing method itself still need a lot of investigations, especially on the training method of the classifier.

Acknowledgement

This project was supported by National Natural Science Foundation of China, Contract 60736014, and 863 State Key Project No. 2006AA010108. We are grateful to the anonymous reviewers for their thorough reviewing and valuable suggestions. We show special thanks to Dr. Rebecca Hwa for generous help of sharing the experimental data. We also thank Dr. Yang Liu for sharing the codes of alignment matrix generation, and Dr. Liang Huang for helpful discussions.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*.

Xavier Carreras, Mihai Surdeanu, and Lluis Marquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the CoNLL*.

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

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the ACL*.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the ICML*, pages 175–182.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP*, pages 1–8, Philadelphia, USA.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the 47th ACL*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the IWPT*, pages 53–64.

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

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the ACL*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. In *Natural Language Engineering*, volume 11, pages 311–325.

Wenbin Jiang and Qun Liu. 2009. Automatic adaptation of annotation standards for dependency parsing using projected treebank as source corpus. In *Proceedings of IWPT*.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging–a case study. In *Proceedings of the 47th ACL*.

Dan Klein and Christopher D. Manning. 2004. Corpusbased induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the ACL*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the ACL*.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the EMNLP*.

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of the EMNLP*.

Yajuan Lü, Sheng Li, Tiejun Zhao, and Muyun Yang. 2002. Learning chinese bracketing knowledge based on a bilingual language model. In *Proceedings of the COLING*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. In *Computational Linguistics*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the COLING*.

Joakim Nivre, Johan Hall, Jens Nilsson, Gulsen Eryigit, and Svetoslav Marinov. 2006. Labeled pseudoprojective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the ACL*.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the ACL*, pages 160–167.

David Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of the EACL*.

Vladimir N. Vapnik. 1998. Statistical learning theory. In *A Wiley-Interscience Publication*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*.

H Yamada and Y Matsumoto. 2003. Statistical dependency analysis using support vector machines. In *Proceedings of IWPT*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *Proceedings of the ACL*.

Bitext Dependency Parsing with Bilingual Subtree Constraints

Wenliang Chen, Jun'ichi Kazama and Kentaro Torisawa
Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{chenwl, kazama, torisawa}@nict.go.jp

Abstract

This paper proposes a dependency parsing method that uses bilingual constraints to improve the accuracy of parsing bilingual texts (bitexts). In our method, a target-side tree fragment that corresponds to a source-side tree fragment is identified via word alignment and mapping rules that are automatically learned. Then it is verified by checking the subtree list that is collected from large scale automatically parsed data on the target side. Our method, thus, requires gold standard trees only on the source side of a bilingual corpus in the training phase, unlike the joint parsing model, which requires gold standard trees on the both sides. Compared to the re-ordering constraint model, which requires the same training data as ours, our method achieved higher accuracy because of richer bilingual constraints. Experiments on the translated portion of the Chinese Treebank show that our system outperforms monolingual parsers by 2.93 points for Chinese and 1.64 points for English.

1 Introduction

Parsing bilingual texts (bitexts) is crucial for training machine translation systems that rely on syntactic structures on either the source side or the target side, or the both (Ding and Palmer, 2005; Nakazawa et al., 2006). Bitexts could provide more information, which is useful in parsing, than a usual monolingual texts that can be called "bilingual constraints", and we expect to obtain more accurate parsing results that can be effectively used in the training of MT systems. With this motivation, there are several studies aiming at highly accurate bitext parsing (Smith and Smith, 2004; Burkett and Klein, 2008; Huang et al., 2009).

This paper proposes a dependency parsing method, which uses the bilingual constraints that we call *bilingual subtree constraints* and statistics concerning the constraints estimated from large unlabeled monolingual corpora. Basically, a (candidate) dependency subtree in a source-language sentence is mapped to a subtree in the corresponding target-language sentence by using word alignment and mapping rules that are automatically learned. The target subtree is verified by checking the subtree list that is collected from unlabeled sentences in the target language parsed by a usual monolingual parser. The result is used as additional features for the source side dependency parser. In this paper, our task is to improve the source side parser with the help of the translations on the target side.

Many researchers have investigated the use of bilingual constraints for parsing (Burkett and Klein, 2008; Zhao et al., 2009; Huang et al., 2009). For example, Burkett and Klein (2008) show that parsing with joint models on bitexts improves performance on either or both sides. However, their methods require that the training data have tree structures on both sides, which are hard to obtain. Our method only requires dependency annotation on the source side and is much simpler and faster. Huang et al. (2009) proposes a method, bilingual-constrained monolingual parsing, in which a source-language parser is extended to use the re-ordering of words between two sides' sentences as additional information. The input of their method is the source trees with their translation on the target side as ours, which is much easier to obtain than trees on both sides. However, their method does not use any tree structures on

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 21–29,
Uppsala, Sweden, 11-16 July 2010.

the target side that might be useful for ambiguity resolution. Our method achieves much greater improvement because it uses the richer subtree constraints.

Our approach takes the same input as Huang et al. (2009) and exploits the subtree structure on the target side to provide the bilingual constraints. The subtrees are extracted from large-scale auto-parsed monolingual data on the target side. The main problem to be addressed is mapping words on the source side to the target subtree because there are many to many mappings and reordering problems that often occur in translation (Koehn et al., 2003). We use an automatic way for generating mapping rules to solve the problems. Based on the mapping rules, we design a set of features for parsing models. The basic idea is as follows: if the words form a subtree on one side, their corresponding words on the another side will also probably form a subtree.

Experiments on the translated portion of the Chinese Treebank (Xue et al., 2002; Bies et al., 2007) show that our system outperforms state-of-the-art monolingual parsers by 2.93 points for Chinese and 1.64 points for English. The results also show that our system provides higher accuracies than the parser of Huang et al. (2009).

The rest of the paper is organized as follows: Section 2 introduces the motivation of our idea. Section 3 introduces the background of dependency parsing. Section 4 proposes an approach of constructing bilingual subtree constraints. Section 5 explains the experimental results. Finally, in Section 6 we draw conclusions and discuss future work.

2 Motivation

In this section, we use an example to show the idea of using the bilingual subtree constraints to improve parsing performance.

Suppose that we have an input sentence pair as shown in Figure 1, where the source sentence is in English, the target is in Chinese, the dashed undirected links are word alignment links, and the directed links between words indicate that they have a (candidate) dependency relation.

In the English side, it is difficult for a parser to determine the head of word "with" because there is a PP-attachment problem. However, in Chinese it is unambiguous. Therefore, we can use the information on the Chinese side to help disambiguation.

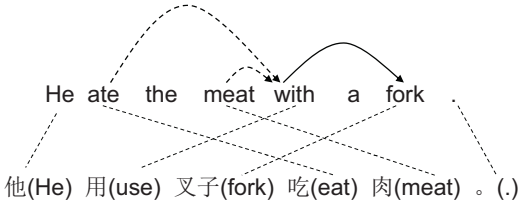


Figure 1: Example for disambiguation

There are two candidates "ate" and "meat" to be the head of "with" as the dashed directed links in Figure 1 show. By adding "fork", we have two possible dependency relations, "meat-with-fork" and "ate-with-fork", to be verified.

First, we check the possible relation of "meat", "with", and "fork". We obtain their corresponding words "肉(meat)", "用(use)", and "叉子(fork)" in Chinese via the word alignment links. We verify that the corresponding words form a subtree by looking up a subtree list in Chinese (described in Section 4.1). But we can not find a subtree for them.

Next, we check the possible relation of "ate", "with", and "fork". We obtain their corresponding words "吃(ate)", "用(use)", and "叉子(fork)". Then we verify that the words form a subtree by looking up the subtree list. This time we can find the subtree as shown in Figure 2.

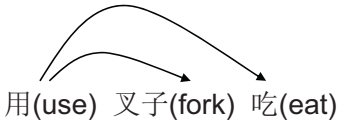


Figure 2: Example for a searched subtree

Finally, the parser may assign "ate" to be the head of "with" based on the verification results. This simple example shows how to use the subtree information on the target side.

3 Dependency parsing

For dependency parsing, there are two main types of parsing models (Nivre and McDonald, 2008; Nivre and Kubler, 2006): transition-based (Nivre, 2003; Yamada and Matsumoto, 2003) and graph-based (McDonald et al., 2005; Carreras, 2007). Our approach can be applied to both parsing models.

In this paper, we employ the graph-based MST parsing model proposed by McDonald and Pereira

(2006), which is an extension of the projective parsing algorithm of Eisner (1996). To use richer second-order information, we also implement parent-child-grandchild features (Carreras, 2007) in the MST parsing algorithm.

3.1 Parsing with monolingual features

Figure 3 shows an example of dependency parsing. In the graph-based parsing model, features are represented for all the possible relations on single edges (two words) or adjacent edges (three words). The parsing algorithm chooses the tree with the highest score in a bottom-up fashion.

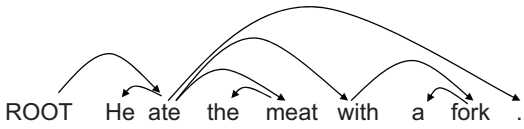


Figure 3: Example of dependency tree

In our systems, the monolingual features include the first- and second- order features presented in (McDonald et al., 2005; McDonald and Pereira, 2006) and the parent-child-grandchild features used in (Carreras, 2007). We call the parser with the monolingual features monolingual parser.

3.2 Parsing with bilingual features

In this paper, we parse source sentences with the help of their translations. A set of bilingual features are designed for the parsing model.

3.2.1 Bilingual subtree features

We design bilingual subtree features, as described in Section 4, based on the constraints between the source subtrees and the target subtrees that are verified by the subtree list on the target side. The source subtrees are from the possible dependency relations.

3.2.2 Bilingual reordering feature

Huang et al. (2009) propose features based on reordering between languages for a shift-reduce parser. They define the features based on word-alignment information to verify that the corresponding words form a contiguous span for resolving shift-reduce conflicts. We also implement similar features in our system.

4 Bilingual subtree constraints

In this section, we propose an approach that uses the bilingual subtree constraints to help parse source sentences that have translations on the target side.

We use large-scale auto-parsed data to obtain subtrees on the target side. Then we generate the mapping rules to map the source subtrees onto the extracted target subtrees. Finally, we design the bilingual subtree features based on the mapping rules for the parsing model. These features indicate the information of the constraints between bilingual subtrees, that are called bilingual subtree constraints.

4.1 Subtree extraction

Chen et al. (2009) propose a simple method to extract subtrees from large-scale monolingual data and use them as features to improve monolingual parsing. Following their method, we parse large unannotated data with a monolingual parser and obtain a set of subtrees (ST_t) in the target language.

We encode the subtrees into string format that is expressed as $st = w : hid(-w : hid)+$[1], where w refers to a word in the subtree and hid refers to the word ID of the word's head (hid=0 means that this word is the root of a subtree). Here, word ID refers to the ID (starting from 1) of a word in the subtree (words are ordered based on the positions of the original sentence). For example, "He" and "ate" have a left dependency arc in the sentence shown in Figure 3. The subtree is encoded as "He:2-ate:0". There is also a parent-child-grandchild relation among "ate", "with", and "fork". So the subtree is encoded as "ate:0-with:1-fork:2". If a subtree contains two nodes, we call it a bigram-subtree. If a subtree contains three nodes, we call it a trigram-subtree.

From the dependency tree of Figure 3, we obtain the subtrees, as shown in Figure 4 and Figure 5. Figure 4 shows the extracted bigram-subtrees and Figure 5 shows the extracted trigram-subtrees. After extraction, we obtain a set of subtrees. We remove the subtrees occurring only once in the data. Following Chen et al. (2009), we also group the subtrees into different sets based on their frequencies.

[1] + refers to matching the preceding element one or more times and is the same as a regular expression in Perl.

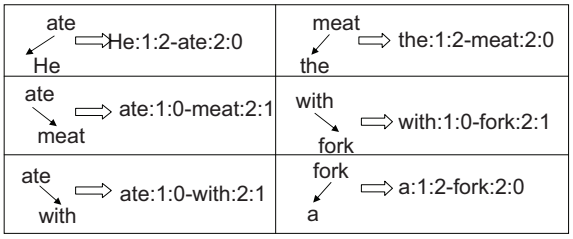


Figure 4: Examples of bigram-subtrees

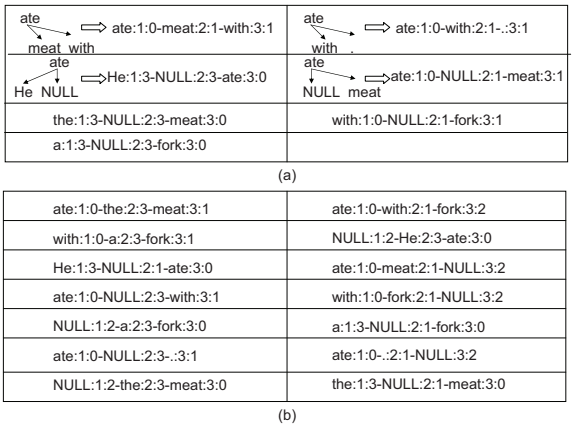


Figure 5: Examples of trigram-subtrees

4.2 Mapping rules

To provide bilingual subtree constraints, we need to find the characteristics of subtree mapping for the two given languages. However, subtree mapping is not easy. There are two main problems: MtoN (words) mapping and reordering, which often occur in translation. MtoN (words) mapping means that a source subtree with M words is mapped onto a target subtree with N words. For example, 2to3 means that a source bigram-subtree is mapped onto a target trigram-subtree.

Due to the limitations of the parsing algorithm (McDonald and Pereira, 2006; Carreras, 2007), we only use bigram- and trigram-subtrees in our approach. We generate the mapping rules for the 2to2, 2to3, 3to3, and 3to2 cases. For trigram-subtrees, we only consider the parent-child-grandchild type. As for the use of other types of trigram-subtrees, we leave it for future work.

We first show the MtoN and reordering problems by using an example in Chinese-English translation. Then we propose a method to automatically generate mapping rules.

4.2.1 Reordering and MtoN mapping in translation

Both Chinese and English are classified as SVO languages because verbs precede objects in simple sentences. However, Chinese has many characteristics of such SOV languages as Japanese. The typical cases are listed below:

1) Prepositional phrases modifying a verb precede the verb. Figure 6 shows an example. In English the prepositional phrase “at the ceremony” follows the verb “said”, while its corresponding prepositional phrase “在(NULL) 仪式(ceremony) 上(at)” precedes the verb “说(say)” in Chinese.

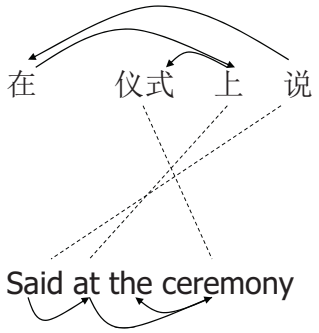


Figure 6: Example for prepositional phrases modifying a verb

2) Relative clauses precede head noun. Figure 7 shows an example. In Chinese the relative clause “今天(today) 签字(signed)” precedes the head noun “项目(project)”, while its corresponding clause “signed today” follows the head noun “projects” in English.

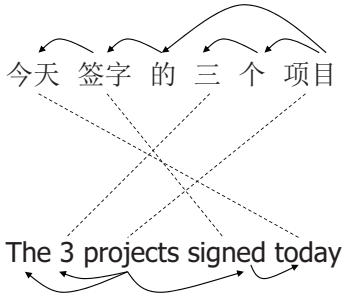


Figure 7: Example for relative clauses preceding the head noun

3) Genitive constructions precede head noun. For example, “汽车(car) 轮子(wheel)” can be translated as “the wheel of the car”.

4) Postposition in many constructions rather than prepositions. For example, “桌子(table) 上(on)” can be translated as “on the table”.

We can find the MtoN mapping problem occurring in the above cases. For example, in Figure 6, trigram-subtree "在(NULL):3-上(at):1-说(say):0" is mapped onto bigram-subtree "said:0-at:1".

Since asking linguists to define the mapping rules is very expensive, we propose a simple method to easily obtain the mapping rules.

4.2.2 Bilingual subtree mapping

To solve the mapping problems, we use a bilingual corpus, which includes sentence pairs, to automatically generate the mapping rules. First, the sentence pairs are parsed by monolingual parsers on both sides. Then we perform word alignment using a word-level aligner (Liang et al., 2006; DeNero and Klein, 2007). Figure 8 shows an example of a processed sentence pair that has tree structures on both sides and word alignment links.

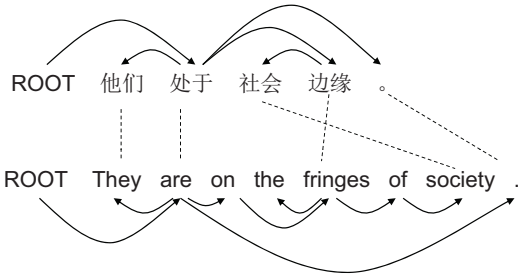


Figure 8: Example of auto-parsed bilingual sentence pair

From these sentence pairs, we obtain subtree pairs. First, we extract a subtree (st_s) from a source sentence. Then through word alignment links, we obtain the corresponding words of the words of st_s. Because of the MtoN problem, some words lack of corresponding words in the target sentence. Here, our approach requires that at least two words of st_s have corresponding words and nouns and verbs need corresponding words. If not, it fails to find a subtree pair for st_s. If the corresponding words form a subtree (st_t) in the target sentence, st_s and st_t are a subtree pair. We also keep the word alignment information in the target subtree. For example, we extract subtree "社会(society):2-边缘(fringe):0" on the Chinese side and get its corresponding subtree "fringes(W_2):0-of:1-society(W_1):2" on the English side, where W_1 means that the target word is aligned to the first word of the source subtree, and W_2 means that the target word is aligned to the second word of the source subtree. That is, we have a subtree pair: "社会(society):2-边缘(fringe):0" and "fringe(W_2):0-of:1-society(W_1):2".

The extracted subtree pairs indicate the translation characteristics between Chinese and English. For example, the pair "社会(society):2-边缘(fringe):0" and "fringes:0-of:1-society:2" is a case where "Genitive constructions precede/follow the head noun".

4.2.3 Generalized mapping rules

To increase the mapping coverage, we generalize the mapping rules from the extracted subtree pairs by using the following procedure. The rules are divided by "=>" into two parts: source (left) and target (right). The source part is from the source subtree and the target part is from the target subtree. For the source part, we replace *nouns* and *verbs* using their POS tags (coarse grained tags). For the target part, we use the word alignment information to represent the target words that have corresponding source words. For example, we have the subtree pair: "社会(society):2-边缘(fringe):0" and "fringes(W_2):0-of:1-society(W_1):2", where "of" does not have a corresponding word, the POS tag of "社会(society)" is N, and the POS tag of "边缘(fringe)" is N. The source part of the rule becomes "N:2-N:0" and the target part becomes "W_2:0-of:1-W_1:2".

Table 1 shows the top five mapping rules of all four types ordered by their frequencies, where W_1 means that the target word is aligned to the first word of the source subtree, W_2 means that the target word is aligned to the second word, and W_3 means that the target word is aligned to the third word. We remove the rules that occur less than three times. Finally, we obtain 9,134 rules for 2to2, 5,335 for 2to3, 7,450 for 3to3, and 1,244 for 3to2 from our data. After experiments with different threshold settings on the development data sets, we use the top 20 rules for each type in our experiments.

The generalized mapping rules might generate incorrect target subtrees. However, as described in Section 4.3.1, the generated subtrees are verified by looking up list ST_t before they are used in the parsing models.

4.3 Bilingual subtree features

Informally, if the words form a subtree on the source side, then the corresponding words on the target side will also probably form a subtree. For

#	rules	freq
2to2 mapping		
1	N:2 N:0 => W_1:2 W_2:0	92776
2	V:0 N:1 => W_1:0 W_2:1	62437
3	V:0 V:1 => W_1:0 W_2:1	49633
4	N:2 V:0 => W_1:2 W_2:0	43999
5	的:2 N:0 => W_2:0 W_1:2	25301
2to3 mapping		
1	N:2-N:0 => W_2:0-of:1-W_1:2	10361
2	V:0-N:1 => W_1:0-of:1-W_2:2	4521
3	V:0-N:1 => W_1:0-to:1-W_2:2	2917
4	N:2-V:0 => W_2:0-of:1-W_1:2	2578
5	N:2-N:0 => W_1:2-':3-W_2:0	2316
3to2 mapping		
1	V:2-的/DEC:3-N:0 => W_1:0-W_3:1	873
2	V:2-的/DEC:3-N:0 => W_3:2-W_1:0	634
3	N:2-的/DEG:3-N:0 => W_1:0-W_3:1	319
4	N:2-的/DEG:3-N:0 => W_3:2-W_1:0	301
5	V:0-的/DEG:3-N:1 => W_3:0-W_1:1	247
3to3 mapping		
1	V:0-V:1-N:2 => W_1:0-W_2:1-W_3:2	9580
2	N:2-的/DEG:3-N:0 => W_3:0-W_2:1-W_1:2	7010
3	V:0-N:3-N:1 => W_1:0-W_2:3-W_3:1	5642
4	V:0-V:1-V:2 => W_1:0-W_2:1-W_3:2	4563
5	N:2-N:3-N:0 => W_1:2-W_2:3-W_3:0	3570

Table 1: Top five mapping rules of 2to3 and 3to2

example, in Figure 8, words "他 们(they)" and "处 于(be_on)" form a subtree , which is mapped onto the words "they" and "are" on the target side. These two target words form a subtree. We now develop this idea as bilingual subtree features.

In the parsing process, we build relations for two or three words on the source side. The conditions of generating bilingual subtree features are that at least two of these source words must have corresponding words on the target side and nouns and verbs must have corresponding words.

At first, we have a possible dependency relation (represented as a source subtree) of words to be verified. Then we obtain the corresponding target subtree based on the mapping rules. Finally, we verify that the target subtree is included in ST_t. If yes, we activate a positive feature to encourage the dependency relation.

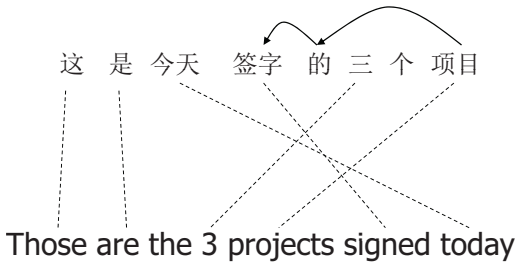


Figure 9: Example of features for parsing

We consider four types of features based on 2to2, 3to3, 3to2, and 2to3 mappings. In the 2to2, 3to3, and 3to2 cases, the target subtrees do not add new words. We represent features in a direct way. For the 2to3 case, we represent features using a different strategy.

4.3.1 Features for 2to2, 3to3, and 3to2

We design the features based on the mapping rules of 2to2, 3to3, and 3to2. For example, we design features for a 3to2 case from Figure 9. The possible relation to be verified forms source subtree "签 字(signed)/VV:2-的(NULL)/DEC:3-项 目(project)/NN:0" in which "项 目(project)" is aligned to "projects" and "签 字(signed)" is aligned to "signed" as shown in Figure 9. The procedure of generating the features is shown in Figure 10. We explain Steps (1), (2), (3), and (4) as follows:

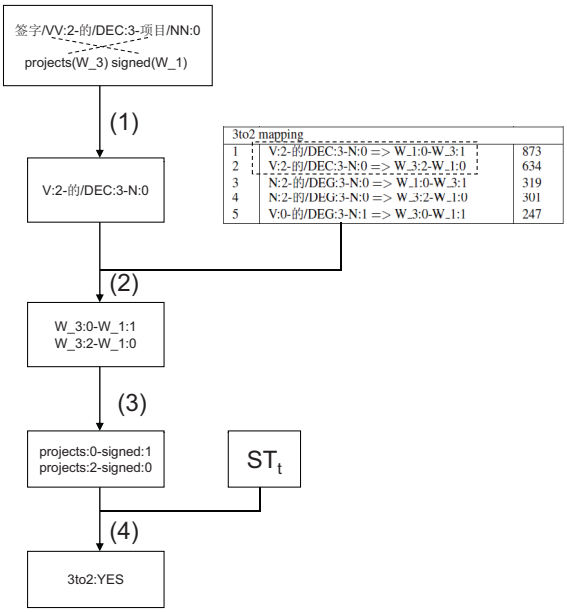


Figure 10: Example of feature generation for 3to2 case

(1) Generate source part from the source subtree. We obtain "V:2-的/DEC:3-N:0" from "签 字(signed)/VV:2-的(NULL)/DEC:3-项 目(project)/NN:0".

(2) Obtain target parts based on the matched mapping rules, whose source parts equal "V:2-的/DEC:3-N:0". The matched rules are "V:2-的/DEC:3-N:0 =>W_3:0-W_1:1" and "V:2-的/DEC:3-N:0 => W_3:2-W_1:0". Thus, we have two target parts "W_3:0-W_1:1" and "W_3:2-W_1:0".

(3) Generate possible subtrees by consider-

ing the dependency relation indicated in the target parts. We generate a possible subtree "projects:0-signed:1" from the target part "W_3:0-W_1:1", where "projects" is aligned to "项目(project)(W_3)" and "signed" is aligned to "签字(signed)(W_1)". We also generate another possible subtree "projects:2-signed:0" from "W_3:2-W_1:0".

(4) Verify that at least one of the generated possible subtrees is a target subtree, which is included in ST_t. If yes, we activate this feature. In the figure, "projects:0-signed:1" is a target subtree in ST_t. So we activate the feature "3to2:YES" to encourage dependency relations among "签字(signed)", "的(NULL)", and "项目(project)".

4.3.2 Features for 2to3

In the 2to3 case, a new word is added on the target side. The first two steps are identical as those in the previous section. For example, a source part "N:2-N:0" is generated from "汽车(car)/NN:2-轮子(wheel)/NN:0". Then we obtain target parts such as "W_2:0-of/IN:1-W_1:2", "W_2:0-in/IN:1-W_1:2", and so on, according to the matched mapping rules.

The third step is different. In the target parts, there is an added word. We first check if the added word is in the span of the corresponding words, which can be obtained through word alignment links. We can find that "of" is in the span "wheel of the car", which is the span of the corresponding words of "汽 车(car)/NN:2-轮 子(wheel)/NN:0". Then we choose the target part "W_2:0-of/IN:1-W_1:2" to generate a possible subtree. Finally, we verify that the subtree is a target subtree included in ST_t. If yes, we say feature "2to3:YES" to encourage a dependency relation between "汽车(car)" and "轮子(wheel)".

4.4 Source subtree features

Chen et al. (2009) shows that the source subtree features (F_{src-st}) significantly improve performance. The subtrees are obtained from the auto-parsed data on the source side. Then they are used to verify the possible dependency relations among source words.

In our approach, we also use the same source subtree features described in Chen et al. (2009). So the possible dependency relations are verified by the source and target subtrees. Combining two types of features together provides strong discrimination power. If both types of features are active, building relations is very likely among source words. If both are inactive, this is a strong negative signal for their relations.

5 Experiments

All the bilingual data were taken from the translated portion of the Chinese Treebank (CTB) (Xue et al., 2002; Bies et al., 2007), articles 1-325 of CTB, which have English translations with gold-standard parse trees. We used the tool "Penn2Malt"[2] to convert the data into dependency structures. Following the study of Huang et al. (2009), we used the same split of this data: 1-270 for training, 301-325 for development, and 271-300 for test. Note that some sentence pairs were removed because they are not one-to-one aligned at the sentence level (Burkett and Klein, 2008; Huang et al., 2009). Word alignments were generated from the Berkeley Aligner (Liang et al., 2006; DeNero and Klein, 2007) trained on a bilingual corpus having approximately 0.8M sentence pairs. We removed notoriously bad links in {a, an, the}×{的(DE), 了(LE)} following the work of Huang et al. (2009).

For Chinese unannotated data, we used the XIN_CMN portion of Chinese Gigaword Version 2.0 (LDC2009T14) (Huang, 2009), which has approximately 311 million words whose segmentation and POS tags are given. To avoid unfair comparison, we excluded the sentences of the CTB data from the Gigaword data. We discarded the annotations because there are differences in annotation policy between CTB and this corpus. We used the MMA system (Kruengkrai et al., 2009) trained on the training data to perform word segmentation and POS tagging and used the Baseline Parser to parse all the sentences in the data. For English unannotated data, we used the BLLIP corpus that contains about 43 million words of WSJ text. The POS tags were assigned by the MXPOST tagger trained on training data. Then we used the Baseline Parser to parse all the sentences in the data.

We reported the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of tokens (excluding all punctuation tokens) with correct HEADs.

5.1 Main results

The results on the Chinese-source side are shown in Table 2, where "Baseline" refers to the systems

[2] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

with monolingual features, "Baseline2" refers to adding the reordering features to the Baseline, "F_{BI}" refers to adding all the bilingual subtree features to "Baseline2", "F_{src-st}" refers to the monolingual parsing systems with source subtree features, "Order-1" refers to the first-order models, and "Order-2" refers to the second-order models. The results showed that the reordering features yielded an improvement of 0.53 and 0.58 points (UAS) for the first- and second-order models respectively. Then we added four types of bilingual constraint features one by one to "Baseline2". Note that the features based on 3to2 and 3to3 can not be applied to the first-order models, because they only consider single dependencies (bigram). That is, in the first model, F_{BI} only includes the features based on 2to2 and 2to3. The results showed that the systems performed better and better. In total, we obtained an absolute improvement of 0.88 points (UAS) for the first-order model and 1.36 points for the second-order model by adding all the bilingual subtree features. Finally, the system with all the features (OURS) outperformed the Baseline by an absolute improvement of 3.12 points for the first-order model and 2.93 points for the second-order model. The improvements of the final systems (OURS) were significant in McNemar's Test ($p < 10^{-4}$).

	Order-1	Order-2
Baseline	84.35	87.20
Baseline2	84.88	87.78
+2to2	85.08	88.07
+2to3	85.23	88.14
+3to3	–	88.29
+3to2	–	88.56
F_{BI}	85.23(+0.88)	88.56(+1.36)
F_{src-st}	86.54(+2.19)	89.49(+2.29)
OURS	87.47(+3.12)	90.13(+2.93)

Table 2: Dependency parsing results of Chinese-source case

We also conducted experiments on the English-source side. Table 3 shows the results, where abbreviations are the same as in Table 2. As in the Chinese experiments, the parsers with bilingual subtree features outperformed the Baselines. Finally, the systems (OURS) with all the features outperformed the Baselines by 1.30 points for the first-order model and 1.64 for the second-order model. The improvements of the final systems (OURS) were significant in McNemar's Test ($p < 10^{-3}$).

	Order-1	Order-2
Baseline	86.41	87.37
Baseline2	86.86	87.66
+2to2	87.23	87.87
+2to3	87.35	87.96
+3to3	–	88.25
+3to2	–	88.37
F_{BI}	87.35(+0.94)	88.37(+1.00)
F_{src-st}	87.25(+0.84)	88.57(+1.20)
OURS	87.71(+1.30)	89.01(+1.64)

Table 3: Dependency parsing results of English-source case

5.2 Comparative results

Table 4 shows the performance of the system we compared, where Huang2009 refers to the result of Huang et al. (2009). The results showed that our system performed better than Huang2009. Compared with the approach of Huang et al. (2009), our approach used additional large-scale auto-parsed data. We did not compare our system with the joint model of Burkett and Klein (2008) because they reported the results on phrase structures.

	Chinese	English
Huang2009	86.3	87.5
Baseline	87.20	87.37
OURS	90.13	89.01

Table 4: Comparative results

6 Conclusion

We presented an approach using large automatically parsed monolingual data to provide bilingual subtree constraints to improve bitexts parsing. Our approach remains the efficiency of monolingual parsing and exploits the subtree structure on the target side. The experimental results show that the proposed approach is simple yet still provides significant improvements over the baselines in parsing accuracy. The results also show that our systems outperform the system of previous work on the same data.

There are many ways in which this research could be continued. First, we may attempt to apply the bilingual subtree constraints to transition-

based parsing models (Nivre, 2003; Yamada and Matsumoto, 2003). Here, we may design new features for the models. Second, we may apply the proposed method for other language pairs such as Japanese-English and Chinese-Japanese. Third, larger unannotated data can be used to improve the performance further.

References

Ann Bies, Martha Palmer, Justin Mott, and Colin Warner. 2007. English Chinese translation treebank v 1.0. In *LDC2007T02*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 877–886, Honolulu, Hawaii, October. Association for Computational Linguistics.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

WL. Chen, J. Kazama, K. Uchimoto, and K. Torisawa. 2009. Improving dependency parsing with subtrees from auto-parsed data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 570–579, Singapore, August. Association for Computational Linguistics.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 541–548, Morristown, NJ, USA. Association for Computational Linguistics.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1222–1231, Singapore, August. Association for Computational Linguistics.

Chu-Ren Huang. 2009. Tagged Chinese Gigaword Version 2.0, LDC2009T14. *Linguistic Data Consortium*.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, page 54. Association for Computational Linguistics.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint Chinese word segmentation and POS tagging. In *Proceedings of ACL-IJCNLP2009*, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL 2005*.

T. Nakazawa, K. Yu, D. Kawahara, and S. Kurohashi. 2006. Example-based machine translation based on deeper nlp. In *Proceedings of IWSLT 2006*, pages 64–70, Kyoto, Japan.

J. Nivre and S. Kubler. 2006. Dependency parsing: Tutorial at Coling-ACL 2006. In *CoLING-ACL*.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT2003*, pages 149–160.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP*.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated Chinese corpus. In *Coling*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT2003*, pages 195–206.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL-IJCNLP2009*, pages 55–63, Suntec, Singapore, August. Association for Computational Linguistics.

Computing weakest readings

Alexander Koller
Cluster of Excellence
Saarland University
koller@mmci.uni-saarland.de

Stefan Thater
Dept. of Computational Linguistics
Saarland University
stth@coli.uni-saarland.de

Abstract

We present an efficient algorithm for computing the weakest readings of semantically ambiguous sentences. A corpus-based evaluation with a large-scale grammar shows that our algorithm reduces over 80% of sentences to one or two readings, in negligible runtime, and thus makes it possible to work with semantic representations derived by deep large-scale grammars.

1 Introduction

Over the past few years, there has been considerable progress in the ability of manually created large-scale grammars, such as the English Resource Grammar (ERG, Copestake and Flickinger (2000)) or the ParGram grammars (Butt et al., 2002), to parse wide-coverage text and assign it deep semantic representations. While applications should benefit from these very precise semantic representations, their usefulness is limited by the presence of semantic ambiguity: On the Rondane Treebank (Oepen et al., 2002), the ERG computes an average of several million semantic representations for each sentence, even when the syntactic analysis is fixed. The problem of appropriately selecting one of them to work with would ideally be solved by statistical methods (Higgins and Sadock, 2003) or knowledge-based inferences. However, no such approach has been worked out in sufficient detail to support the disambiguation of treebank sentences.

As an alternative, Bos (2008) proposes to compute the *weakest* reading of each sentence and then use it instead of the "true" reading of the sentence. This is based on the observation that the readings of a semantically ambiguous sentence are partially ordered with respect to logical entailment, and the weakest readings – the minimal (least informative) readings with respect to this order – only express "safe" information that is common to all other readings as well. However, when a sentence has millions of readings, finding the weakest reading is a hard problem. It is of course completely infeasible to compute all readings and compare all pairs for entailment; but even the best known algorithm in the literature (Gabsdil and Striegnitz, 1999) is only an optimization of this basic strategy, and would take months to compute the weakest readings for the sentences in the Rondane Treebank.

In this paper, we propose a new, efficient approach to the problem of computing weakest readings. We follow an *underspecification* approach to managing ambiguity: Rather than deriving all semantic representations from the syntactic analysis, we work with a single, compact underspecified semantic representation, from which the semantic representations can then be extracted by need. We then approximate entailment with a rewrite system that rewrites readings into logically weaker readings; the weakest readings are exactly those readings that cannot be rewritten into some other reading any more (the *relative normal forms*). We present an algorithm that computes the relative normal forms, and evaluate it on the underspecified descriptions that the ERG derives on a 624-sentence subcorpus of the Rondane Treebank. While the mean number of scope readings in the subcorpus is in the millions, our system computes on average 4.5 weakest readings for each sentence, in less than twenty milliseconds; over 80% of all sentences are reduced to at most two weakest readings. In other words, we make it feasible for the first time to build an application that uses the individual (weakest) semantic representations computed by the ERG, both in terms of the remaining ambiguity and in terms of performance. Our technique is not limited to the ERG, but should be applicable to other underspecification-based grammars as well.

Technically, we use underspecified descriptions that are regular tree grammars derived from dominance graphs (Althaus et al., 2003; Koller et al.,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 30–39,
Uppsala, Sweden, 11-16 July 2010.

2008). We compute the weakest readings by intersecting these grammars with other grammars representing the rewrite rules. This approach can be used much more generally than just for the computation of weakest readings; we illustrate this by showing how a more general version of the redundancy elimination algorithm by Koller et al. (2008) can be seen as a special case of our construction. Thus our system can serve as a general framework for removing unintended readings from an underspecified representation.

The paper is structured as follows. Section 2 starts by reviewing related work. We recall dominance graphs, regular tree grammars, and the basic ideas of underspecification in Section 3, before we show how to compute weakest readings (Section 4) and logical equivalences (Section 5). In Section 6, we define a weakening rewrite system for the ERG and evaluate it on the Rondane Treebank. Section 7 concludes and points to future work.

2 Related work

The idea of deriving a single approximative semantic representation for ambiguous sentences goes back to Hobbs (1983); however, Hobbs only works his algorithm out for a restricted class of quantifiers, and his representations can be weaker than our weakest readings. Rules that weaken one reading into another were popular in the 1990s underspecification literature (Reyle, 1995; Monz and de Rijke, 2001; van Deemter, 1996) because they simplify logical reasoning with underspecified representations. From a linguistic perspective, Kempson and Cormack (1981) even go so far as to claim that the weakest reading should be taken as the "basic" reading of a sentence, and the other readings only seen as pragmatically licensed special cases.

The work presented here is related to other approaches that reduce the set of readings of an underspecified semantic representation (USR). Koller and Niehren (2000) showed how to strengthen a dominance constraint using information about anaphoric accessibility; later, Koller et al. (2008) presented and evaluated an algorithm for redundancy elimination, which removes readings from an USR based on logical equivalence. Our system generalizes the latter approach and applies it to a new inference problem (weakest readings) which they could not solve.

This paper builds closely upon Koller and Thater (2010), which lays the formal groundwork for the work presented here. Here we go beyond that paper by applying a concrete implementation of our RTG construction for weakest readings to a real-world grammar, evaluating the system on practical inputs, and combining weakest readings with redundancy elimination.

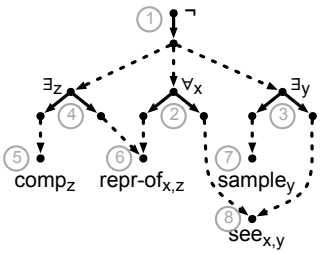


Figure 1: A dominance graph describing the five readings of the sentence "it is not the case that every representative of a company saw a sample."

3 Underspecification

This section briefly reviews two formalisms for specifying sets of trees: dominance graphs and regular tree grammars. Both of these formalisms can be used to model scope ambiguities compactly by regarding the semantic representations of a sentence as trees. Some example trees are shown in Fig. 2. These trees can be read as simplified formulas of predicate logic, or as formulas involving generalized quantifiers (Barwise and Cooper, 1981). Formally, we assume a ranked signature Σ of tree constructors $\{f, g, a, \ldots\}$, each of which is equipped with an arity $\mathsf{ar}(f) \geq 0$. We take a *(finite constructor) tree t* as a finite tree in which each node is labelled with a symbol of Σ, and the number of children of the node is exactly the arity of this symbol. For instance, the signature of the trees in Fig. 1 is $\{\forall_x|2, \exists_y|2, comp_z|0, \ldots\}$. Finite constructor trees can be seen as ground terms over Σ that respect the arities. We write $T(\Sigma)$ for the finite constructor trees over Σ.

3.1 Dominance graphs

A (labelled) dominance graph D (Althaus et al., 2003) is a directed graph that consists of a collection of trees called *fragments*, plus *dominance edges* relating nodes in different fragments. We distinguish the *roots* W_D of the fragments from their *holes*, which are the unlabelled leaves. We write $L_D : W_D \to \Sigma$ for the labeling function of D.

The basic idea behind using dominance graphs to model scope underspecification is to specify

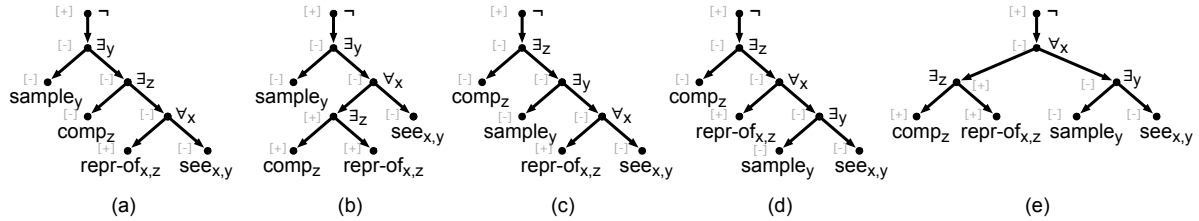


Figure 2: The five configurations of the dominance graph in Fig. 1.

the "semantic material" common to all readings as fragments, plus dominance relations between these fragments. An example dominance graph D is shown in Fig. 1. It represents the five readings of the sentence "it is not the case that every representative of a company saw a sample."

Each reading is encoded as a *(labeled) configuration* of the dominance graph, which can be obtained by "plugging" the tree fragments into each other, in a way that respects the dominance edges: The source node of each dominance edge must dominate (be an ancestor of) the target node in each configuration. The trees in Fig. 2 are the five labeled configurations of the example graph.

3.2 Regular tree grammars

Regular tree grammars (RTGs) are a general grammar formalism for describing languages of trees (Comon et al., 2007). An RTG is a 4-tuple $G = (S, N, \Sigma, P)$, where N and Σ are nonterminal and terminal alphabets, $S \in N$ is the start symbol, and P is a finite set of production rules. Unlike in context-free string grammars (which look superficially the same), the terminal symbols are tree constructors from Σ. The production rules are of the form $A \to t$, where A is a nonterminal and t is a tree from $T(\Sigma \cup N)$; nonterminals count as having arity zero, i.e. they must label leaves. A derivation starts with a tree containing a single node labeled with S. Then in each step of the derivation, some leaf u which is labelled with a nonterminal A is expanded with a rule $A \to t$; this results in a new tree in which u has been replaced by t, and the derivation proceeds with this new tree. The *language* $L(G)$ generated by the grammar is the set of all trees in $T(\Sigma)$ that can be derived in this way.

Fig. 3 shows an RTG as an example. This grammar uses sets of root names from D as nonterminal symbols, and generates exactly the five configurations of the graph in Fig. 1.

The languages that can be accepted by regular tree grammars are called *regular tree languages*

$$
\begin{aligned}
\{1,2,3,4,5,6,7,8\} &\to \neg(\{2,3,4,5,6,7,8\}) \\
\{2,3,4,5,6,7,8\} &\to \forall_x(\{4,5,6\},\{3,7,8\}) \\
\{2,3,4,5,6,7,8\} &\to \exists_y(\{7\},\{2,4,5,6,8\}) \\
\{2,3,4,5,6,7,8\} &\to \exists_z(\{5\},\{2,3,6,7,8\}) \\
\{2,4,5,6,8\} &\to \forall_x(\{4,5,6\},\{8\}) \\
&\quad \mid \exists_z(\{5\},\{2,6,8\}) \\
\{2,3,6,7,8\} &\to \forall_x(\{6\},\{3,7,8\}) \\
&\quad \mid \exists_y(\{7\},\{2,6,8\}) \\
\{2,6,8\} &\to \forall_x(\{6\},\{8\}) \\
\{3,7,8\} &\to \exists_y(\{7\},\{8\}) \\
\{4,5,6\} &\to \exists_z(\{5\},\{6\}) \\
\{5\} \to \mathit{comp}_z \qquad & \{7\} \to \mathit{sample}_y \\
\{6\} \to \mathit{repr\text{-}of}_{x,z} \qquad & \{8\} \to \mathit{see}_{x,y}
\end{aligned}
$$

Figure 3: A regular tree grammar that generates the five trees in Fig. 2.

(RTLs), and regular tree grammars are equivalent to *finite tree automata*, which are defined essentially like the well-known finite string automata, except that they assign states to the nodes in a tree rather than the positions in a string. Regular tree languages enjoy many of the closure properties of regular string languages. In particular, we will later exploit that RTLs are closed under intersection and complement.

3.3 Dominance graphs as RTGs

An important class of dominance graphs are *hypernormally connected (hnc)* dominance graphs (Koller et al., 2003). The precise definition of hnc graphs is not important here, but note that virtually all underspecified descriptions that are produced by current grammars are hypernormally connected (Flickinger et al., 2005), and we will restrict ourselves to hnc graphs for the rest of the paper.

Every hypernormally connected dominance graph D can be automatically translated into an equivalent RTG G_D that generates exactly the same configurations (Koller et al., 2008); the RTG in Fig. 3 is an example. The nonterminals of G_D are

always hnc subgraphs of D. In the worst case, G_D can be exponentially bigger than D, but in practice it turns out that the grammar size remains manageable: even the RTG for the most ambiguous sentence in the Rondane Treebank, which has about 4.5×10^{12} scope readings, has only about 75 000 rules and can be computed in a few seconds.

4 Computing weakest readings

Now we are ready to talk about computing the weakest readings of a hypernormally connected dominance graph. We will first explain how we approximate logical weakening with rewrite systems. We will then discuss how weakest readings can be computed efficiently as the *relative normal forms* of these rewrite systems.

4.1 Weakening rewrite systems

The different readings of a sentence with a scope ambiguity are not a random collection of formulas; they are partially ordered with respect to logical entailment, and are structurally related in a way that allows us to model this entailment relation with simpler technical means.

To illustrate this, consider the five configurations in Fig. 2. The formula represented by (d) logically entails (c); we say that (c) is a *weaker* reading than (d) because it is satisfied by more models. Similar entailment relations hold between (d) and (e), (e) and (b), and so on (see also Fig. 5). We can define the *weakest readings* of the dominance graph as the minimal elements of the entailment order; in the example, these are (b) and (c). Weakest readings capture "safe" information in that whichever reading of the sentence the speaker had in mind, any model of this reading also satisfies at least one weakest reading; in the absence of convincing disambiguation methods, they can therefore serve as a practical approximation of the intended meaning of the sentence.

A naive algorithm for computing weakest readings would explicitly compute the entailment order, by running a theorem prover on each pair of configurations, and then pick out the minimal elements. But this algorithm is quadratic in the number of configurations, and therefore impractically slow for real-life sentences.

Here we develop a fast algorithm for this problem. The fundamental insight we exploit is that entailment among the configurations of a dominance graph can be approximated with rewriting rules (Baader and Nipkow, 1999). Consider the relation between (d) and (c). We can explain that (d) entails (c) by observing that (c) can be built from (d) by exchanging the positions of the adjacent quantifiers $\forall_x$ and $\exists_y$; more precisely, by applying the following rewrite rule:

$$[-]\ \forall_x(Q, \exists_y(P,R)) \to \exists_y(P, \forall_x(Q,R)) \qquad (1)$$

The body of the rule specifies that an occurrence of $\forall_x$ which is the direct parent of an occurrence of $\exists_y$ may change positions with it; the subformulas P, Q, and R must be copied appropriately. The annotation $[-]$ specifies that we must only apply the rule to subformulas in negative logical polarity: If the quantifiers in (d) were not in the scope of a negation, then applying the rule would actually make the formula *stronger*. We say that the rule (1) is logically *sound* because applying it to a subformula with the correct polarity of some configuration t always makes the result t' logically weaker than t.

We formalize these rewrite systems as follows. We assume a finite *annotation alphabet* Ann with a special *starting annotation* $\mathsf{a}_0 \in \mathsf{Ann}$; in the example, we had $\mathsf{Ann} = \{+,-\}$ and $\mathsf{a}_0 = +$. We also assume an *annotator function* $\mathsf{ann} : \mathsf{Ann} \times \Sigma \times \mathbb{N} \to \mathsf{Ann}$. The function ann can be used to traverse a tree top-down and compute the annotation of each node from the annotation of its parent: Its first argument is the annotation and its second argument the node label of the parent, and the third argument is the position of the child among the parent's children. In our example, the annotator ann models logical polarity by mapping, for instance, $\mathsf{ann}(+,\exists_z,1) = \mathsf{ann}(+,\exists_z,2) = \mathsf{ann}(+,\exists_y,2) = +$, $\mathsf{ann}(-,\exists_z,1) = \mathsf{ann}(-,\exists_z,2) = \mathsf{ann}(+,\forall_x,1) = -$, etc. We have labelled each node of the configurations in Fig. 1 with the annotations that are computed in this way.

Now we can define an *annotated rewrite system* R to be a finite set of pairs (a,r) where a is an annotation and r is an ordinary rewrite rule. The rule (1) above is an example of an annotated rewrite rule with $a = -$. A rewrite rule (a,r) can be applied at the node u of a tree t if ann assigns the annotation a to u and r is applicable at u as usual. The rule then rewrites t as described above. In other words, annotated rewrite systems are rewrite systems where rule applications are restricted to subtrees with specific annotations. We write $t \to_R t'$ if some rule of R can be applied at a node of t, and the result of rewriting is t'. The rewrite system R is called *linear*

if every variable that occurs on the left-hand side of a rule occurs on its right-hand side exactly once.

4.2 Relative normal forms

The rewrite steps of a sound weakening rewrite system are related to the entailment order: Because every rewrite step transforms a reading into a weaker reading, an actual weakest readings must be such that there is no other configuration into which it can be rewritten. The converse is not always true, i.e. there can be non-rewritable configurations that are not weakest readings, but we will see in Section 6 that this approximation is good enough for practical use. So one way to solve the problem of computing weakest readings is to find readings that cannot be rewritten further.

One class of configurations that "cannot be rewritten" with a rewrite system R is the set of *normal forms* of R, i.e. those configurations to which no rule in R can be applied. In our example, (b) and (c) are indeed normal forms with respect to a rewrite system that consists only of the rule (1). However, this is not exactly what we need here. Consider a rewrite system that also contains the following annotated rewrite rule, which is also sound for logical entailment:

$$[+]\ \neg(\exists_z(P,Q)) \to \exists_z(P,\neg(Q)), \qquad (2)$$

This rule would allow us to rewrite the configuration (c) into the tree $\exists_z(\mathsf{comp}_z, \neg(\exists_y(\mathsf{sample}_y, \forall_x(\mathsf{repr{-}of}_{x,z}, \mathsf{see}_{x,y}))))$. But this is no longer a configuration of the graph. If we were to equate weakest readings with normal forms, we would erroneously classify (c) as not being a weakest reading. The correct concept for characterizing weakest readings in terms of rewriting is that of a *relative normal form*. We define a configuration t of a dominance graph D to be a R-relative normal form of (the configurations of) D iff there is no other configuration t' of D such that $t \to_R t'$. These are the configurations that can't be weakened further without obtaining a tree that is no longer a configuration of D. In other words, if R approximates entailment, then the R-relative normal forms approximate the weakest readings.

4.3 Computing relative normal forms

We now show how the relative normal forms of a dominance graph can be computed efficiently. For lack of space, we only sketch the construction and omit all proofs. Details can be found in Koller and Thater (2010).

The key idea of the construction is to represent the relation $\to_R$ in terms of a *context tree transducer* M, and characterize the relative normal forms of a tree language L in terms of the pre-image of L under M. Like ordinary regular tree transducers (Comon et al., 2007), context tree transducers read an input tree, assigning states to the nodes, while emitting an output tree. But while ordinary transducers read the input tree symbol by symbol, a context tree transducer can read multiple symbols at once. In this way, they are equivalent to the extended left-hand side transducers of Graehl et al. (2008).

We will now define context tree transducers. Let Σ be a ranked signature, and let X_m be a set of m variables. We write $\mathrm{Con}^{(m)}(\Sigma)$ for the *contexts* with m holes, i.e. those trees in $T(\Sigma \cup X_m)$ in which each element of X_m occurs exactly once, and always as a leaf. If $C \in \mathrm{Con}^{(m)}(\Sigma)$, then $C[t_1,\ldots,t_m] = C[t_1/x_1,\ldots,t_m/x_m]$, where $x_1,\ldots,x_m$ are the variables from left to right.

A *(top-down) context tree transducer* from Σ to Δ is a 5-tuple $M = (Q,\Sigma,\Delta,q_0,\delta)$. Σ and Δ are ranked signatures, Q is a finite set of states, and $q_0 \in Q$ is the start state. δ is a finite set of transition rules of the form $q(C[x_1,\ldots,x_n]) \to D[q_1(x_{i_1}),\ldots,q_m(x_{i_m})]$, where $C \in \mathrm{Con}^{(n)}(\Sigma)$ and $D \in \mathrm{Con}^{(m)}(\Delta)$.

If $t \in T(\Sigma \cup \Delta \cup Q)$, then we say that M derives t' in one step from t, $t \to_M t'$, if t is of the form $C'[q(C[t_1,\ldots,t_n])]$ for some $C' \in \mathrm{Con}^{(1)}(\Sigma)$, t' is of the form $C'[D[q_1(t_{i_1}),\ldots,q_m(t_{i_m})]]$, and there is a rule $q(C[x_1,\ldots,x_n]) \to D[q_1(x_{i_1}),\ldots,q_m(x_{i_m})]$ in δ. The *derivation relation* $\to_M^*$ is the reflexive, transitive closure of $\to_M$. The *translation relation* τ_M of M is

$$\tau_M = \{(t,t') \mid t \in T(\Sigma) \text{ and } t' \in T(\Delta) \text{ and } q_0(t) \to^* t'\}.$$

For each linear annotated rewrite system R, we can now build a context tree transducer M_R such that $t \to_R t'$ iff $(t,t') \in \tau_{M_R}$. The idea is that M_R traverses t from the root to the leaves, keeping track of the current annotation in its state. M_R can nondeterministically choose to either copy the current symbol to the output tree unchanged, or to apply a rewrite rule from R. The rules are built in such a way that in each run, exactly one rewrite rule must be applied.

We achieve this as follows. M_R takes as its states the set $\{\bar{q}\} \cup \{q^a \mid a \in \mathsf{Ann}\}$ and as its start state the state q^{a_0}. If M_R reads a node u in state q^a, this means that the annotator assigns annotation a to u and M_R will rewrite a subtree at or

below u. If M_R reads u in state $\bar{q}$, this means that M_R will copy the subtree below u unchanged because the rewriting has taken place elsewhere. Thus M_R has three types of rewrite rules. First, for any $f \in \Sigma$, we have a rule $\bar{q}(f(x_1,\ldots,x_n)) \to f(\bar{q}(x_1),\ldots,\bar{q}(x_n))$. Second, for any f and $1 \leq i \leq n$, we have a rule $q^a(f(x_1,\ldots,x_n)) \to f(\bar{q}(x_1),\ldots,q^{\mathsf{ann}(a,f,i)}(x_i),\ldots,\bar{q}(x_n))$, which nondeterministically chooses under which child the rewriting should take place, and assigns it the correct annotation. Finally, we have a rule $q^a(C[x_1,\ldots,x_n]) \to C'[\bar{q}(x_{i_1}),\ldots,\bar{q}(x_{i_n})]$ for every rewrite rule $C[x_1,\ldots,x_n] \to C'[x_{i_1},\ldots,x_{i_n}]$ with annotation a in R.

Now let's put the different parts together. We know that for each hnc dominance graph D, there is a regular tree grammar G_D such that $L(G_D)$ is the set of configurations of D. Furthermore, the pre-image $\tau_M^{-1}(L) = \{t \mid \text{exists } t' \in L \text{ with } (t,t') \in \tau_M\}$ of a regular tree language L is also regular (Koller and Thater, 2010) if M is linear, and regular tree languages are closed under intersection and complement (Comon et al., 2007). So we can compute another RTG G' such that

$$L(G') = L(G_D) \cap \overline{\tau_{M_R}^{-1}(L(G_D))}.$$

$L(G')$ consists of the members of $L(G_D)$ which cannot be rewritten by M_R into members of $L(G_D)$; that is, $L(G')$ is exactly the set of R-relative normal forms of D. In general, the complement construction requires exponential time in the size of M_R and G_D. However, it can be shown that if the rules in R have at most depth two and G_D is deterministic, then the entire above construction can be computed in time $O(|G_D| \cdot |R|)$ (Koller and Thater, 2010).

In other words, we have shown how to compute the weakest readings of a hypernormally connected dominance graph D, as approximated by a weakening rewrite system R, in time linear in the size of G_D and linear in the size of R. This is a dramatic improvement over the best previous algorithm, which was quadratic in $|\mathsf{conf}(D)|$.

4.4 An example

Consider an annotated rewrite system that contains rule (1) plus the following rewrite rule:

$$[-]\ \exists_z(P, \forall_x(Q,R)) \to \forall_x(\exists_z(P,Q),R) \qquad (3)$$

This rewrite system translates into a top-down context tree transducer M_R with the following transition rules, omitting most rules of the first two types for lack of space.

$$\begin{aligned}
q^-(\forall_x(x_1,\exists_y(x_2,x_3))) &\to \exists_y(\bar{q}(x_2),\forall_x(\bar{q}(x_1),\bar{q}(x_3)))\\
q^-(\exists_y(x_1,\forall_x(x_2,x_3))) &\to \forall_x(\exists_y(\bar{q}(x_1),\bar{q}(x_2)),\bar{q}(x_3))\\
\bar{q}(\neg(x_1)) &\to \neg(\bar{q}(x_1))\\
q^+(\neg(x_1)) &\to \neg(q^-(x_1))\\
\bar{q}(\forall_x(x_1,x_2)) &\to \forall_x(\bar{q}(x_1),\bar{q}(x_2))\\
q^+(\forall_x(x_1,x_2)) &\to \forall_x(\bar{q}(x_1),q^+(x_2))\\
q^+(\forall_x(x_1,x_2)) &\to \forall_x(q^-(x_1),\bar{q}(x_2)) \quad \ldots
\end{aligned}$$

$$\begin{aligned}
\{1,2,3,4,5,6,7,8\}_F &\to \neg(\{2,3,4,5,6,7,8\}_F)\\
\{2,3,4,5,6,7,8\}_F &\to \exists_y(\{7\}_{\{\bar{q}\}},\{2,4,5,6,8\}_F)\\
&\quad\ \mid \exists_z(\{5\}_{\{\bar{q}\}},\{2,3,6,7,8\}_F)\\
\{2,3,6,7,8\}_F &\to \exists_y(\{7\}_{\{\bar{q}\}},\forall_x(\{6\}_{\{\bar{q}\}},\{8\}_{\{\bar{q}\}}))\\
\{2,4,5,6,8\}_F &\to \forall_x(\{4,5,6\}_{\{\bar{q}\}},\{8\}_{\{\bar{q}\}})\\
\{4,5,6\}_{\{\bar{q}\}} &\to \exists_z(\{5\}_{\{\bar{q}\}},\{6\}_{\{\bar{q}\}})\\
\{5\}_{\{\bar{q}\}} &\to \mathit{comp}_z \qquad \{6\}_{\{\bar{q}\}} \to \mathit{repr\text{-}of}_{x,z}\\
\{7\}_{\{\bar{q}\}} &\to \mathit{sample}_y \quad \{8\}_{\{\bar{q}\}} \to \mathit{see}_{x,y}
\end{aligned}$$

Figure 4: RTG for the weakest readings of Fig. 1.

The grammar G' for the relative normal forms is shown in Fig. 4 (omitting rules that involve unproductive nonterminals). We obtain it by starting with the example grammar G_D in Fig. 3; then computing a deterministic RTG G_R for $\tau_{M_R}^{-1}(L(G_D))$; and then intersecting the complement of G_R with G_D. The nonterminals of G' are subgraphs of D, marked either with a set of states of M_R or the symbol F, indicating that G_R had no production rule for a given left-hand side. The start symbol of G' is marked with F because G' should only generate trees that G_R can*not* generate. As expected, G' generates precisely two trees, namely (b) and (c).

5 Redundancy elimination, revisited

The construction we just carried out – characterize the configurations we find interesting as the relative normal forms of an annotated rewrite system R, translate it into a transducer M_R, and intersect $\mathsf{conf}(D)$ with the complement of the pre-image under M_R – is more generally useful than just for the computation of weakest readings. We illustrate this on the problem of *redundancy elimination* (Vestre, 1991; Chaves, 2003; Koller et al., 2008) by showing how a variant of the algorithm of Koller et al. (2008) falls out of our technique as a special case.

Redundancy elimination is the problem of computing, from a dominance graph D, another dominance graph D' such that $\mathsf{conf}(D') \subseteq \mathsf{conf}(D)$ and

every formula in $\mathsf{conf}(D)$ is *logically equivalent* to some formula in $\mathsf{conf}(D')$. We can approximate logical equivalence using a finite system of equations such as

$$\exists_y(P, \exists_z(Q,R)) = \exists_z(Q, \exists_y(P,R)), \qquad (4)$$

indicating that $\exists_y$ and $\exists_z$ can be permuted without changing the models of the formula.

Following the approach of Section 4, we can solve the redundancy elimination problem by transforming the equation system into a rewrite system R such that $t \to_R t'$ implies that t and t' are equivalent. To this end, we assume an arbitrary linear order $<$ on Σ, and orient all equations into rewrite rules that respect this order. If we assume $\exists_y < \exists_z$, the example rule (4) translates into the annotated rewrite rules

$$[a]\ \exists_z(P, \exists_y(Q,R)) \to \exists_y(Q, \exists_z(P,R)) \qquad (5)$$

for all annotations $a \in \mathsf{Ann}$; logical equivalence is not sensitive to the annotation. Finally, we can compute the relative normal forms of $\mathsf{conf}(D)$ under this rewrite system as above. The result will be an RTG G' describing a subset of $\mathsf{conf}(D)$. Every tree t in $\mathsf{conf}(D)$ that is not in $L(G')$ is equivalent to some tree t' in $L(G')$, because if t could not be rewritten into such a t', then t would be in relative normal form. That is, the algorithm solves the redundancy elimination problem. Furthermore, if the oriented rewrite system is confluent (Baader and Nipkow, 1999), no two trees in $L(G')$ will be equivalent to each other, i.e. we achieve complete reduction in the sense of Koller et al. (2008).

This solution shares much with that of Koller et al. (2008), in that we perform redundancy elimination by intersecting tree grammars. However, the construction we present here is much more general: The algorithmic foundation for redundancy elimination is now exactly the same as that for weakest readings, we only have to use an equivalence-preserving rewrite system instead of a weakening one. This new formal clarity also simplifies the specification of certain equations, as we will see in Section 6.

In addition, we can now combine the weakening rules (1), (3), and (5) into a single rewrite system, and then construct a tree grammar for the relative normal forms of the combined system. This algorithm performs redundancy elimination and computes weakest readings at the same time, and in our example retains only a single configuration, namely

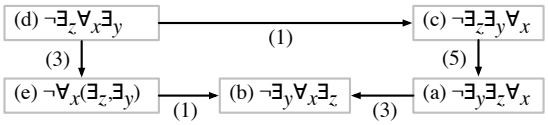


Figure 5: Structure of the configuration set of Fig. 1 in terms of rewriting.

(b); the configuration (c) is rejected because it can be rewritten to (a) with (5). The graph in Fig. 5 illustrates how the equivalence and weakening rules conspire to exclude all other configurations.

6 Evaluation

In this section, we evaluate the effectiveness and efficiency of our weakest readings algorithm on a treebank. We compute RTGs for all sentences in the treebank and measure how many weakest readings remain after the intersection, and how much time this computation takes.

Resources. For our experiment, we use the Rondane treebank (version of January 2006), a "Redwoods style" (Oepen et al., 2002) treebank containing underspecified representations (USRs) in the MRS formalism (Copestake et al., 2005) for sentences from the tourism domain.

Our implementation of the relative normal forms algorithm is based on Utool (Koller and Thater, 2005), which (among other things) can translate a large class of MRS descriptions into hypernormally connected dominance graphs and further into RTGs as in Section 3. The implementation exploits certain properties of RTGs computed from dominance graphs to maximize efficiency. We will make this implementation publically available as part of the next Utool release.

We use Utool to automatically translate the 999 MRS descriptions for which this is possible into RTGs. To simplify the specification of the rewrite systems, we restrict ourselves to the subcorpus in which all scope-taking operators (labels with arity > 0) occur at least ten times. This subset contains 624 dominance graphs. We refer to this subset as "RON10."

Signature and annotations. For each dominance graph D that we obtain by converting an MRS description, we take G_D as a grammar over the signature $\Sigma = \{f_u \mid u \in W_D, f = L_D(u)\}$. That is, we distinguish possible different occurrences of the same symbol in D by marking each occur-

rence with the name of the node. This makes G_D a deterministic grammar.

We then specify an annotator over Σ that assigns polarities for the weakening rewrite system. We distinguish three polarities: $+$ for positive occurrences, $-$ for negative occurrences (as in predicate logic), and $\perp$ for contexts in which a weakening rule neither weakens or strengthens the entire formula. The starting annotation is $+$.

Finally, we need to decide upon each scope-taking operator's effects on these annotations. To this end, we build upon Barwise and Cooper's (1981) classification of the monotonicity properties of determiners. A determiner is *upward (downward) monotonic* if making the denotation of the determiner's argument bigger (smaller) makes the sentence logically weaker. For instance, *every* is downward monotonic in its first argument and upward monotonic in its second argument, i.e. *every girl kissed a boy* entails *every blond girl kissed someone*. Thus $\mathsf{ann}(\mathsf{every}_u, a, 1) = -a$ and $\mathsf{ann}(\mathsf{every}_u, a, 2) = a$ (where u is a node name as above). There are also determiners with non-monotonic argument positions, which assign the annotation $\perp$ to this argument. Negation reverses positive and negative polarity, and all other non-quantifiers simply pass on their annotation to the arguments.

Weakest readings. We use the following weakening rewrite system for our experiment, where $i \in \{1,2\}$:

1. [+] $(\mathsf{E}/i, \mathsf{D}/1)$, $(\mathsf{D}/2, \mathsf{D}/1)$
2. [+] $(\mathsf{E}/i, \mathsf{P}/1)$, $(\mathsf{D}/2, \mathsf{P}/1)$
3. [+] $(\mathsf{E}/i, \mathsf{A}/2)$, $(\mathsf{D}/1, \mathsf{A}/2)$
4. [+] $(\mathsf{A}/2, \mathsf{N}/1)$
5. [+] $(\mathsf{N}/1, \mathsf{E}/i)$, $(\mathsf{N}/1, \mathsf{D}/2)$
6. [+] $(\mathsf{E}/i, \mathsf{M}/1)$, $(\mathsf{D}/1, \mathsf{M}/1)$

Here the symbols E, D, etc. stand for classes of labels in Σ, and a rule schema $[a]$ $(\mathsf{C}/i, \mathsf{C}'/k)$ is to be read as shorthand for a set of rewrite rules which rearrange a tree where the i-th child of a symbol from C is a symbol from C' into a tree where the symbol from C becomes the k-th child of the symbol from C'. For example, because we have $\mathsf{all}_u \in \mathsf{A}$ and $\mathsf{not}_v \in \mathsf{N}$, Schema 4 licenses the following annotated rewrite rule:

$$[+]\ \mathsf{all}_u(P, \mathsf{not}_v(Q)) \rightarrow \mathsf{not}_v(\mathsf{all}_u(P, Q)).$$

We write E and D for existential and definite determiners. P stands for proper names and pronouns, A stands for universal determiners like *all* and *each*, N for the negation *not,* and M for modal operators like *can* or *would*. M also includes intensional verbs like *have to* and *want*. Notice that while the reverse rules are applicable in negative polarities, no rules are applicable in polarity $\perp$.

Rule schema 1 states, for instance, that the specific (wide-scope) reading of the indefinite in *the president of a company* is logically stronger than the reading in which *a company* is within the restriction of the definite determiner. The schema is intuitively plausible, and it can also be proved to be logically sound if we make the standard assumption that the definite determiner *the* means "exactly one" (Montague, 1974). A similar argument applies to rule schema 2.

Rule schema 3 encodes the classical entailment (1). Schema 4 is similar to the rule (2). Notice that it is not, strictly speaking, logically sound; however, because strong determiners like *all* or *every* carry a presupposition that their restrictions have a non-empty denotation (Lasersohn, 1993), the schema becomes sound for all instances that can be expressed in natural language. Similar arguments apply to rule schemas 5 and 6, which are potentially unsound for subtle reasons involving the logical interpretation of intensional expressions. However, these cases of unsoundness did not occur in our test corpus.

Redundancy elimination. In addition, we assume the following equation system for redundancy elimination for $i, j \in \{1,2\}$ and $k \in \mathbb{N}$ (again written in an analogous shorthand as above):

7. $\mathsf{E}/i = \mathsf{E}/j$
8. $\mathsf{D}/1 = \mathsf{E}/i,\ \mathsf{E}/i = \mathsf{D}/1$
9. $\mathsf{D}/1 = \mathsf{D}/1$
10. $\Sigma/k = \mathsf{P}/2$

These rule schemata state that permuting existential determiners with each other is an equivalence transformation, and so is permuting definite determiners with existential and definite determiners if one determiner is the second argument (in the scope) of a definite. Schema 10 states that proper names and pronouns, which the ERG analyzes as scope-bearing operators, can permute with any other label.

We orient these equalities into rewrite rules by ordering symbols in P before symbols that are not

	All	KRT08	RE	RE+WR
#conf = 1	8.5%	23.4%	34.9%	66.7%
#conf ≤ 2	20.5%	40.9%	57.9%	80.6%
avg(#conf)	3.2M	7603.1	119.0	4.5
med(#conf)	25	4	2	1
runtime	8.1s	9.4s	8.7s	9.1s

Figure 6: Analysis of the numbers of configurations in RON10.

in P, and otherwise ordering a symbol f_u before a symbol g_v if $u < v$ by comparison of the (arbitrary) node names.

Results. We used these rewrite systems to compute, for each USR in RON10, the number of all configurations, the number of configurations that remain after redundancy elimination, and the number of weakest readings (i.e., the relative normal forms of the combined equivalence and weakening rewrite systems). The results are summarized in Fig. 6. By computing weakest readings (WR), we reduce the ambiguity of over 80% of all sentences to one or two readings; this is a clear improvement even over the results of the redundancy elimination (RE). Computing weakest readings reduces the mean number of readings from several million to 4.5, and improves over the RE results by a factor of 30. Notice that the RE algorithm from Section 5 is itself an improvement over Koller et al.'s (2008) system ("KRT08" in the table), which could not process the rule schema 10.

Finally, computing the weakest readings takes only a tiny amount of extra runtime compared to the RE elimination or even the computation of the RTGs (reported as the runtime for "All").[1] This remains true on the entire Rondane corpus (although the reduction factor is lower because we have no rules for the rare scope-bearers): RE+WR computation takes 32 seconds, compared to 30 seconds for RE. In other words, our algorithm brings the semantic ambiguity in the Rondane Treebank down to practically useful levels at a mean runtime investment of a few milliseconds per sentence.

It is interesting to note how the different rule schemas contribute to this reduction. While the instances of Schemata 1 and 2 are applicable in 340 sentences, the other schemas 3–6 together are only applicable in 44 sentences. Nevertheless, where these rules do apply, they have a noticeable effect: Without them, the mean number of configurations in RON10 after RE+WR increases to 12.5.

[1] Runtimes were measured on an Intel Core 2 Duo CPU at 2.8 GHz, under MacOS X 10.5.6 and Apple Java 1.5.0_16, after allowing the JVM to just-in-time compile the bytecode.

7 Conclusion

In this paper, we have shown how to compute the weakest readings of a dominance graph, characterized by an annotated rewrite system. Evaluating our algorithm on a subcorpus of the Rondane Treebank, we reduced the mean number of configurations of a sentence from several million to 4.5, in negligible runtime. Our algorithm can be applied to other problems in which an underspecified representation is to be disambiguated, as long as the remaining readings can be characterized as the relative normal forms of a linear annotated rewrite system. We illustrated this for the case of redundancy elimination.

The algorithm presented here makes it possible, for the first time, to derive a single meaningful semantic representation from the syntactic analysis of a deep grammar on a large scale. In the future, it will be interesting to explore how these semantic representations can be used in applications. For instance, it seems straightforward to adapt MacCartney and Manning's (2008) "natural logic"-based Textual Entailment system, because our annotator already computes the polarities needed for their monotonicity inferences. We could then perform such inferences on (cleaner) semantic representations, rather than strings (as they do).

On the other hand, it may be possible to reduce the set of readings even further. We retain more readings than necessary in many treebank sentences because the combined weakening and equivalence rewrite system is not confluent, and therefore may not recognize a logical relation between two configurations. The rewrite system could be made more powerful by running the Knuth-Bendix completion algorithm (Knuth and Bendix, 1970). Exploring the practical tradeoff between the further reduction in the number of remaining configurations and the increase in complexity of the rewrite system and the RTG would be worthwhile.

Acknowledgments. We are indebted to Joachim Niehren, who pointed out a crucial simplification in the algorithm to us. We also thank our reviewers for their constructive comments.

References

E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48:194–219.

F. Baader and T. Nipkow. 1999. *Term rewriting and all that*. Cambridge University Press.

J. Barwise and R. Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

J. Bos. 2008. Let's not argue about semantics. In *Proceedings of the 6th international conference on Language Resources and Evaluation (LREC 2008)*.

M. Butt, H. Dyvik, T. Holloway King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*.

R. P. Chaves. 2003. Non-redundant scope disambiguation in underspecified semantics. In *Proceedings of the 8th ESSLLI Student Session*.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`.

A. Copestake and D. Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*.

A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2005. Minimal recursion semantics: An introduction. *Journal of Language and Computation*.

D. Flickinger, A. Koller, and S. Thater. 2005. A new well-formedness criterion for semantics debugging. In *Proceedings of the 12th International Conference on HPSG*, Lisbon.

M. Gabsdil and K. Striegnitz. 1999. Classifying scope ambiguities. In *Proceedings of the First Intl. Workshop on Inference in Computational Semantics*.

J. Graehl, K. Knight, and J. May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

D. Higgins and J. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1).

J. Hobbs. 1983. An improper treatment of quantification in ordinary English. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL'83)*.

R. Kempson and A. Cormack. 1981. Ambiguity and quantification. *Linguistics and Philosophy*, 4:259–309.

D. Knuth and P. Bendix. 1970. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford.

A. Koller and J. Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*.

A. Koller and S. Thater. 2005. Efficient solving and exploration of scope ambiguities. In *ACL-05 Demonstration Notes*, Ann Arbor.

A. Koller and S. Thater. 2010. Computing relative normal forms in regular tree languages. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA)*.

A. Koller, J. Niehren, and S. Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Proceedings of the 10th EACL*.

A. Koller, M. Regneri, and S. Thater. 2008. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of ACL-08: HLT*.

P. Lasersohn. 1993. Existence presuppositions and background knowledge. *Journal of Semantics*, 10:113–122.

B. MacCartney and C. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*.

R. Montague. 1974. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press, New Haven.

C. Monz and M. de Rijke. 2001. Deductions with meaning. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics, Third International Conference (LACL'98)*, volume 2014 of *LNAI*. Springer-Verlag, Berlin/Heidelberg.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*.

Uwe Reyle. 1995. On reasoning with ambiguities. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*.

K. van Deemter. 1996. Towards a logic of ambiguous expressions. In *Semantic Ambiguity and Underspecification*. CSLI Publications, Stanford.

E. Vestre. 1991. An algorithm for generating non-redundant quantifier scopings. In *Proc. of EACL*, Berlin.

Identifying Generic Noun Phrases

Nils Reiter and **Anette Frank**
Department of Computational Linguistics
Heidelberg University, Germany
{reiter,frank}@cl.uni-heidelberg.de

Abstract

This paper presents a supervised approach for identifying generic noun phrases in context. Generic statements express rule-like knowledge about kinds or events. Therefore, their identification is important for the automatic construction of knowledge bases. In particular, the distinction between generic and non-generic statements is crucial for the correct encoding of generic and instance-level information. Generic expressions have been studied extensively in formal semantics. Building on this work, we explore a corpus-based learning approach for identifying generic NPs, using selections of linguistically motivated features. Our results perform well above the baseline and existing prior work.

1 Introduction

Generic expressions come in two basic forms: generic noun phrases and generic sentences. Both express rule-like knowledge, but in different ways.

A **generic noun phrase** is a noun phrase that does not refer to a specific (set of) individual(s), but rather to a kind or class of individuals. Thus, the NP *The lion* in (1.a)[1] is understood as a reference to the class "lion" instead of a specific individual. Generic NPs are not restricted to occur with kind-related predicates as in (1.a). As seen in (1.b), they may equally well be combined with predicates that denote specific actions. In contrast to (1.a), the property defined by the verb phrase in (1.b) may hold of individual lions.

(1) a. The lion was the most widespread mammal.

b. Lions eat up to 30 kg in one sitting.

[1] All examples are taken from Wikipedia unless stated otherwise.

Generic sentences are characterising sentences that quantify over situations or events, expressing rule-like knowledge about habitual actions or situations (2.a). This is in contrast with sentences that refer to specific events and individuals, as in (2.b).

(2) a. After 1971 [Paul Erdős] also took amphetamines.

b. Paul Erdős was born [...] on March 26, 1913.

The genericity of an expression may arise from the generic (kind-referring, class-denoting) interpretation of the NP or the characterising interpretation of the sentence predicate. Both sources may concur in a single sentence, as illustrated in Table 1, where we have cross-classified the examples above according to the genericity of the NP and the sentence.

This classification is extremely difficult, because (i) the criteria for generic interpretation are far from being clear-cut and (ii) both sources of genericity may freely interact.

	S[gen+]	S[gen-]
NP[gen+]	(1.b)	(1.a)
NP[gen-]	(2.a)	(2.b)

Table 1: Generic NPs and generic sentences

The above classification of generic expressions is well established in traditional formal semantics (cf. Krifka et al. (1995))[2]. As we argue in this paper, these distinctions are relevant for semantic processing in computational linguistics, especially for information extraction and ontology learning and population tasks. With appropriate semantic analysis of generic statements, we can not only formally capture and exploit generic knowledge,

[2] The literature draws some finer distinctions including aspects like specificity, which we will ignore in this work.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 40–49,
Uppsala, Sweden, 11-16 July 2010.

but also distinguish between information pertaining to individuals vs. classes. We will argue that the automatic identification of generic expressions should be cast as a machine learning problem instead of a rule-based approach, as there is (i) no transparent marking of genericity in English (as in most other European languages) and (ii) the phenomenon is highly context dependent.

In this paper, we build on insights from formal semantics to establish a corpus-based machine learning approach for the automatic classification of generic expressions. In principle our approach is applicable to the detection of both generic NPs and generic sentences, and in fact it would be highly desirable and possibly advantageous to cover both types of genericity simultaneously. Our current work is confined to generic NPs, as there are no corpora available at present that contain annotations for genericity at the sentence level.

The paper is organised as follows. Section 2 introduces generic expressions and motivates their relevance for knowledge acquisition and semantic processing tasks in computational linguistics. Section 3 reviews prior and related work. In section 4 we motivate the choice of feature sets for the automatic identification of generic NPs in context. Sections 5 and 6 present our experiments and results obtained for this task on the ACE-2 data set. Section 7 concludes.

2 Generic Expressions & their Relevance for Computational Linguistics

2.1 Interpretation of generic expressions

Generic NPs There are two contrasting views on how to formally interpret generic NPs. According to the first one, a generic NP involves a special form of **quantification**. Quine (1960), for example, proposes a universally quantified reading for generic NPs. This view is confronted with the most important problem of all quantification-based approaches, namely that the exact determination of the quantifier restriction (QR) is highly dependent on the context, as illustrated in (3)[3].

(3) a. Lions are mammals. QR: *all lions*

b. Mammals give birth to live young. QR: *less than half of all mammals*

c. Rats are bothersome to people. QR: *few rats*[4]

In view of this difficulty, several approaches restrict the quantification to only "relevant" (Declerck, 1991) or "normal" (Dahl, 1975) individuals.

According to the second view, generic noun phrases denote **kinds**. Following Carlson (1977), a kind can be considered as an individual that has properties on its own. On this view, the generic NP cannot be analysed as a quantifier over individuals pertaining to the kind. For some predicates, this is clearly marked. (1.a), for instance, attributes a property to the kind lion that cannot be attributed to individual lions.

Generic sentences are usually analysed using a special dyadic operator, as first proposed by Heim (1982). The dyadic operator relates two semantic constituents, the *restrictor* and the *matrix*:

$$Q[x_1, ..., x_i](\underbrace{[x1, ..., x_i]}_{\textit{Restrictor}}; \underbrace{\exists y_1, ..., y_i[x_1, .., x_i, y_1, ..., y_i]}_{\textit{Matrix}})$$

By choosing GEN as a generic dyadic operator, it is possible to represent the two readings (a) and (b) of the characterising sentence (4) by variation in the specification of restrictor and matrix (Krifka et al., 1995).

(4) Typhoons arise in this part of the pacific.

(a) Typhoons in general have a common origin in this part of the pacific.

(b) There arise typhoons in this part of the pacific.

(a') GEN$[x; y]$(Typhoon(x);this-part-of-the-pacific$(y) \wedge$arise-in(x, y))

(b') GEN$[x; y]$(this-part-of-the-pacific(x);Typhoon$(y) \wedge$arise-in(y, x))

In order to cope with characterising sentences as in (2.a), we must allow the generic operator to quantify over situations or events, in this case, "normal" situations which were such that Erdős took amphetamines.

2.2 Relevance for computational linguistics

Knowledge acquisition The automatic acquisition of formal knowledge for computational applications is a major endeavour in current research

[3] Some of these examples are taken from Carlson (1977).

[4] Most rats are not even noticed by people.

and could lead to big improvements of semantics-based processing. Bos (2009), e.g., describes systems using automated deduction for language understanding tasks using formal knowledge.

There are manually built formal ontologies such as SUMO (Niles and Pease, 2001) or Cyc (Lenat, 1995) and linguistic ontologies like WordNet (Fellbaum, 1998) that capture linguistic and world knowledge to a certain extent. However, these resources either lack coverage or depth. Automatically constructed ontologies or taxonomies, on the other hand, are still of poor quality (Cimiano, 2006; Ponzetto and Strube, 2007).

Attempts to automatically induce knowledge bases from text or encyclopaedic sources are currently not concerned with the distinction between generic and non-generic expressions, concentrating mainly on factual knowledge. However, rule-like knowledge can be found in textual sources in the form of generic expressions[5].

In view of the properties of generic expressions discussed above, this lack of attention bears two types of risks. The first concerns the distinction between classes and instances, regarding the attribution of properties. The second concerns modelling exceptions in both representation and inferencing.

The distinction between **classes and instances** is a serious challenge even for the simplest methods in automatic ontology construction, e.g., Hearst (1992) patterns. The so-called IS-A patterns do not only identify subclasses, but also instances. *Shakespeare*, e.g., would be recognised as a hyponym of *author* in the same way as *temple* is recognised as a hyponym of *civic building*.

Such a missing distinction between classes and instances is problematic. First, there are predicates that can only attribute properties to a kind (1.a). Second, even for properties that in principle can be attributed to individuals of the class, this is highly dependent on the selection of the quantifier's restriction in context (3). In both cases, it holds that properties attributed to a class are not necessarily inherited by any or all instances pertaining to the class.

Zirn et al. (2008) are the first to present fully automatic, heuristic methods to distinguish between classes and instances in the Wikipedia taxonomy derived by Ponzetto and Strube (2007). They report an accuracy of 81.6% and 84.5% for different classification schemes. However, apart from a plural feature, all heuristics are tailored to specific properties of the Wikipedia resource.

Modelling exceptions is a cumbersome but necessary problem to be handled in ontology building, be it manually or by automatic means, and whether or not the genericity of knowledge is formalised explicitly. In artificial intelligence research, this area has been tackled for many years. Default reasoning (Reiter, 1980) is confronted with severe efficiency problems and therefore has not extended beyond experimental systems. However, the emerging paradigm of Answer Set Programming (ASP, Lifschitz (2008)) seems to be able to model exceptions efficiently. In ASP a given problem is cast as a logic program, and an answer set solver calculates all possible answer sets, where an answer set corresponds to a solution of the problem. Efficient answer set solvers have been proposed (Gelfond, 2007). Although ASP may provide us with very efficient reasoning systems, it is still necessary to distinguish and mark default rules explicitly (Lifschitz, 2002). Hence, the recognition of generic expressions is an important precondition for the adequate representation and processing of generic knowledge.

3 Prior Work

Suh (2006) applied a rule-based approach to automatically identify generic noun phrases. Suh used patterns based on part of speech tags that identify bare plural noun phrases, reporting a precision of 28.9% for generic entities, measured against an annotated corpus, the ACE 2005 (Ferro et al., 2005). Neither recall nor f-measure are reported. To our knowledge, this is the single prior work on the task of identifying generic NPs.

Next to the ACE corpus (described in more detail below), Herbelot and Copestake (2008) offer a study on annotating genericity in a corpus. Two annotators annotated 48 noun phrases from the British National Corpus for their genericity (and specificity) properties, obtaining a kappa value of 0.744. Herbelot and Copestake (2008) leave su-

[5] In the field of cognitive science, research on the acquisition of generic knowledge in humans has shown that adult speakers tend to use generic expressions very often when talking to children (Pappas and Gelman, 1998). We are not aware of any detailed assessment of the proportion of generic noun phrases in educational text genres or encyclopaedic resources like Wikipedia. Concerning generic sentences, Mathew and Katz (2009) report that 19.9% of the sentences in their annotated portion of the Penn Treebank are habitual (generic) and 80.1% episodic (non-generic).

pervised learning for the identification of generic expressions as future work.

Recent work by Mathew and Katz (2009) presents automatic classification of generic and non-generic sentences, yet restricted to habitual interpretations of generic sentences. They use a manually annotated part of the Penn TreeBank as training and evaluation set[6]. Using a selection of syntactic and semantic features operating mainly on the sentence level, they achieved precision between 81.2% and 84.3% and recall between 60.6% and 62.7% for the identification of habitual generic sentences.

4 Characterising Generic Expressions for Automatic Classification

4.1 Properties of generic expressions

Generic NPs come in various syntactic forms. These include definite and indefinite singular count nouns, bare plural count and singular and plural mass nouns as in (5.a-f). (5.f) shows a construction that makes the kind reading unambiguous. As Carlson (1977) observed, the generic reading of "well-established" kinds seems to be more prominent (g vs. h).

(5) a. The lion was the most widespread mammal.

b. A lioness is weaker [...] than a male.

c. Lions died out in northern Eurasia.

d. Metals are good conductors.

e. Metal is also used for heat sinks.

f. The zoo has one kind of tiger.

g. The Coke bottle has a narrow neck.

h. The green bottle has a narrow neck.

Apart from being all NPs, there is no obvious syntactic property that is shared by all examples. Similarly, generic sentences come in a range of syntactic forms (6).

(6) a. John walks to work.

b. John walked to work (when he lived in California).

c. John will walk to work (when he moves to California).

[6]The corpus has not been released.

Although generic NPs and generic sentences can be combined freely (cf. Section 1; Table 1), both phenomena highly interact and quite often appear in the same sentence (Krifka et al., 1995). Also, genericity is highly dependent on contextual factors. Present tense, e.g., may be indicative for genericity, but with appropriate temporal modification, generic sentences may occur in past or future tense (6). Presence of a copular construction as in (5.a,b,d) may indicate a generic NP reading, but again we find generic NPs with event verbs, as in (5.e) or (1.b). Lexical semantic factors, such as the semantic type of the clause predicate (5.c,e), or "well-established" kinds (5.g) may favour a generic reading, but such lexical factors are difficult to capture in a rule-based setting.

In our view, these observations call for a corpus-based machine learning approach that is able to capture a variety of factors indicating genericity in combination and in context.

4.2 Feature set and feature classes

In Table 2 we give basic information about the individual features we investigate for identifying generic NPs. In the following, we will structure this feature space along two dimensions, distinguishing NP- and sentence-level factors as well as syntactic and semantic (including lexical semantic) factors. Table 3 displays the grouping into corresponding feature classes.

NP-level features are extracted from the local NP without consideration of the sentence context.

Sentence-level features are extracted from the clause (in which the NP appears), as well as sentential and non-sentential adjuncts of the clause. We also included the (dependency) relations between the target NP and its governing clause.

Syntactic features are extracted from a parse tree or shallow surface-level features. The feature set includes NP-local and global features.

Semantic features include semantic features abstracted from syntax, such as tense and aspect or type of modification, but also lexical semantic features such as word sense classes, sense granularity or verbal predicates.

Our aim is to determine indicators for genericity from combinations of these feature classes.

Feature	Description
Number	sg, pl
Person	1, 2, 3
Countability	ambig, no noun, count, uncount
Noun Type	common, proper, pronoun
Determiner Type	def, indef, demon
Granularity	The number of edges in the WordNet hypernymy graph between the synset of the entity and a top node
Part of Speech	POS-tag (Penn TreeBank tagset; Marcus et al. (1993)) of the head of the phrase
Bare Plural	false, true
Sense[0-3]	WordNet sense. Sense[0] represents the sense of the head of the entity, Sense[1] its direct hypernym sense and so forth.
Sense[Top]	The top sense in the hypernym hierarchy (often referred to as "super sense")
Dependency Relation [0-4]	Dependency Relations. Relation[0] represents the relation between entity and its governor, Relation[1] the relation between the governor and its governor and so forth.
Embedded Predicate.Pred	Lemma of the head of the directly governing predicate of the entity
C.Tense	past, pres, fut
C.Progressive	false, true
C.Perfective	false, true
C.Mood	indicative, imperative, subjunctive
C.Passive	false, true
C.Temporal Modifier?	false, true
C.Number of Modifiers	numeric
C.Part of Speech	POS-tag (Penn TreeBank tagset; Marcus et al. (1993)) of the head of the phrase
C.Pred	Lemma of the head of the clause
C.Adjunct.Time	true, false
C.Adjunct.VType	main, copular
C.Adjunct.Adverbial Type	vpadv, sadv
C.Adjunct.Degree	positive, comparative, superlative
C.Adjunct.Pred	Lemma of the head of the adjunct of the clause
XLE.Quality	How complete is the parse by the XLE parser? fragmented, complete, no parse

Table 2: The features used in our system. C stands for the clause in which the noun phrase appears, "Embedding Predicate" its direct predicate. In most cases, we just give the value range, if necessary, we give descriptions. All features may have a NULL value.

	Syntactic	Semantic
NP-level	Number, Person, Part of Speech, Determiner Type, Bare Plural	Countability, Granularity, Sense[0-3, Top]
S-level	Clause.{Part of Speech, Passive, Number of Modifiers}, Dependency Relation[0-4], Clause.Adjunct.{Verbal Type, Adverbial Type}, XLE.Quality	Clause.{Tense, Progressive, Perfective, Mood, Pred, Has temporal Modifier}, Clause.Adjunct.{Time, Pred}, Embedded Predicate.Pred

Table 3: Feature classes

Name	Descriptions and Features
Set 1	Five best single features: Bare Plural, Person, Sense [0], Clause.Pred, Embedding Predicate.Pred
Set 2	Five best feature tuples: a. Number, Part of Speech b. Countability, Part of Speech c. Sense [0], Part of Speech d. Number, Countability e. Noun Type, Part of Speech
Set 3	Five best feature triples: a. Number, Clause.Tense, Part of Speech b. Number, Clause.Tense, Noun Type c. Number, Clause.Part of Speech, Part of Speech d. Number, Part of Speech, Noun Type e. Number, Clause.Part of Speech, Noun Type
Set 4	Features, that appear most often among the single, tuple and triple tests: Number, Noun Type, Part of Speech, Clause.Tense, Clause.Part of Speech, Clause.Pred, Embedding Predicate.Pred, Person, Sense [0], Sense [1], Sense[2]
Set 5	Features performing best in the ablation test: Number, Person, Clause.Part of Speech, Clause.Pred, Embedding Predicate.Pred, Clause.Tense, Determiner Type, Part of Speech, Bare Plural, Dependency Relation [2], Sense [0]

Table 4: Derived feature sets

5 Experiments

5.1 Dataset

As data set we are using the ACE-2 (Mitchell et al., 2003) corpus, a collection of newspaper texts annotated with entities marked for their genericity. In this version of the corpus, the classification of entities is a binary one.

Annotation guidelines The ACE-2 annotation guidelines describe generic NPs as referring to an arbitrary member of the set in question, rather than to a particular individual. Thus, a property attributed to a generic NP is in principle applicable to arbitrary members of the set (although not to all of them). The guidelines list several tests that are either local syntactic tests involving determiners or tests that cannot be operationalised as they involve world knowledge and context information.

The guidelines give a number of criteria to identify generic NPs referring to specific properties. These are (i) types of entities (*lions* in 3.a), (ii) suggested attributes of entities (*mammals* in 3.a), (iii) hypothetical entities (7) and (iv) generalisations across sets of entities (5.d).

(7) If a person steps over the line, they must be punished.

The general description of generic NPs as denoting arbitrary members of sets obviously does not capture kind-referring readings. However, the properties characterised (i) can be understood to admit kinds. Also, some illustrations in the guidelines explicitly characterise kind-referring NPs as generic. Thus, while at first sight the guidelines do not fully correspond to the characterisation of generics we find in the formal semantics literature, we argue that both characterisations have similar extensions, i.e., include largely overlapping sets of noun phrases. In fact, all of the examples for generic noun phrases presented in this paper would also be classified as generic according to the ACE-2 guidelines.

We also find annotated examples of generic NPs that are not discussed in the formal semantics literature (8.a), but that are well captured by the ACE-2 guidelines. However, there are also cases that are questionable (8.b).

(8) a. "It's probably not the perfect world, but you kind of have to deal with what you have to work with," he said.

b. Even more remarkable is the Internet, where information of all kinds is available about the government and the economy.

This shows that the annotation of generics is difficult, but also highlights the potential benefit of a corpus-driven approach that allows us to gather a wider range of realisations. This in turn can contribute to novel insights and discussion.

Data analysis A first investigation of the corpus shows that generic NPs are much less common than non-generic ones, at least in the newspaper genre at hand. Of the 40,106 annotated entities, only 5,303 (13.2%) are marked as generic. In order to control for bias effects in our classifier, we will experiment with two different training sets, a balanced and an unbalanced one.

5.2 Preprocessing

The texts have been (pre-)processed to add several layers of linguistic annotation (Table 5). We use MorphAdorner for sentence splitting and TreeTagger with the standard parameter files for part of speech tagging and lemmatisation. As we do not have a word sense disambiguation system available that outperforms the most frequent sense baseline, we simply used the most frequent sense (MFS). The countability information is taken from Celex. Parsing was done using the English LFG grammar (cf. Butt et al. (2002)) in the XLE parsing platform and the Stanford Parser.

Task	Tool
Sentence splitting	MorphAdorner [7]
POS, lemmatisation	TreeTagger (Schmid, 1994)
WSD	MFS (according to WordNet 3.0)
Countability	Celex (Baayen et al., 1996)
Parsing	XLE (Crouch et al., 2010)
	Stanford (Klein and Manning, 2003)

Table 5: Preprocessing pipeline

As the LFG-grammar produced full parses only for the sentences of 56% of the entities (partial parses: 37% of the entities), we chose to integrate the Stanford parser as a fallback. If we are unable to extract feature values from the f-structure produced by the XLE parser, we extract them from the Stanford Parser, if possible. Experimentation showed using the two parsers in tandem yields best results, compared to individual use.

[7] http://morphadorner.northwestern.edu

Feature Set			Generic			Non generic			Overall		
			P	R	F	P	R	F	P	R	F
		Baseline Majority	0	0	0	86.8	100	92.9	75.3	86.8	80.6
		Baseline *Person*	**60.5**	10.2	17.5	87.9	99.0	93.1	84.3	**87.2**	**85.7**
		Baseline Suh	28.9								
Feature Classes	Unbalanced	NP	31.7	56.6	40.7	92.5	81.4	86.6	84.5	78.2	81.2
		S	32.2	50.7	39.4	91.8	83.7	87.6	83.9	79.4	81.6
		NP/Syntactic	39.2	58.4	46.9	93.2	86.2	89.5	86.0	82.5	84.2
		S/Syntactic	31.9	22.1	26.1	88.7	92.8	90.7	81.2	83.5	82.3
		NP/Semantic	28.2	53.5	36.9	91.8	79.2	85	83.4	75.8	79.4
		S/Semantic	32.1	36.6	34.2	90.1	88.2	89.2	82.5	81.4	81.9
		Syntactic	**40.1**	66.6	**50.1**	94.3	84.8	89.3	**87.2**	82.4	84.7
		Semantic	34.5	56.0	42.7	92.6	83.8	88.0	84.9	80.1	82.4
		All	37.0	**72.1**	49.0	81.3	87.6	87.4	80.1	80.1	83.6
	Balanced	NP	30.1	71.0	42.2	94.4	74.8	83.5	85.9	74.3	79.7
		S	26.9	73.1	39.3	94.4	69.8	80.3	85.5	70.2	77.1
		NP/Syntactic	**35.4**	76.3	**48.4**	95.6	78.8	86.4	87.7	78.5	82.8
		S/Syntactic	23.1	77.1	35.6	94.6	61.0	74.2	85.1	63.1	72.5
		NP/Semantic	24.7	60.0	35.0	92.2	72.1	80.9	83.3	70.5	76.4
		S/Semantic	26.4	66.3	37.7	93.3	71.8	81.2	84.5	71.1	77.2
		Syntactic	30.8	**85.3**	45.3	96.9	70.8	81.9	88.2	72.8	79.7
		Semantic	30.1	67.5	41.6	93.9	76.1	84.1	85.5	75.0	79.9
		All	33.7	81.0	47.6	96.3	75.8	84.8	88.0	76.5	81.8
Feature Selection	Unbalanced	Set 1	**49.5**	37.4	42.6	90.8	94.2	92.5	85.3	86.7	86.0
		Set 2a	37.3	42.7	39.8	91.1	89.1	90.1	84.0	82.9	83.5
		Set 3a	42.6	54.1	47.7	92.7	88.9	90.8	86.1	84.3	85.2
		Set 4	42.7	**69.6**	52.9	94.9	85.8	90.1	88.0	83.6	85.7
		Set 5	45.7	64.8	**53.6**	94.3	88.3	91.2	87.9	85.2	**86.5**
	Balanced	Set 1	29.7	71.1	41.9	94.4	74.4	83.2	85.9	73.9	79.5
		Set 2a	36.5	70.5	48.1	94.8	81.3	87.5	87.1	79.8	83.3
		Set 3a	36.2	70.8	47.9	94.8	81.0	87.4	87.1	79.7	83.2
		Set 4	35.9	**83.1**	50.1	96.8	77.4	86.0	88.7	78.2	83.1
		Set 5	**37.0**	81.9	**51.0**	96.6	78.7	86.8	88.8	79.2	**83.7**

Table 6: Results of the classification, using different feature and training sets

5.3 Experimental setup

Given the unclear dependencies of features, we chose to use a Bayesian network. A Bayesian network represents the dependencies of random variables in a directed acyclic graph, where each node represents a random variable and each edge a dependency between variables. In fact, a number of feature selection tests uncovered feature dependencies (see below). We used the Weka (Witten and Frank, 2002) implementation BayesNet in all our experiments.

To control for bias effects, we created balanced data sets by oversampling the number of generic entities and simultaneously undersampling non-generic entities. This results in a dataset of 20,053 entities with approx. 10,000 entities for each class. All experiments are performed on balanced and unbalanced data sets using 10-fold cross-validation, where balancing has been performed for each training fold separately (if any).

Feature classes We performed evaluation runs for different combinations of feature sets: NP- vs. S-level features (with further distinction between syntactic and semantic NP-/S-level features), as well as overall syntactic vs. semantic features. This was done in order to determine the effect of different types of linguistic factors for the detection of genericity (cf. Table 3).

Feature selection We experimented with two methods for feature selection. Table 4 shows the resulting feature sets.

In **ablation testing**, a single feature in turn is temporarily omitted from the feature set. The feature whose omission causes the biggest drop in f-measure is set aside as a strong feature. This process is repeated until we are left with an empty feature set. From the ranked list of features f_1 to f_n we evaluate increasingly extended feature sets $f_1..f_i$ for i = 2..n. We select the feature set that yields the best balanced performance, at 45.7% precision and 53.6% f-measure. The features are given as Set 5 in Table 4.

As ablation testing does not uncover feature dependencies, we also experimented with **single, tuple** and **triple feature combinations** to determine features that perform well in combination. We ran evaluations using features in isolation and each possible pair and triple of features. We select the resulting five best features, tuples and triples of features. The respective feature sets are given as Set 1 to Set 3 in Table 4. The features that appear most often in Set 1 to Set 3 are grouped in Set 4.

Baseline Our results are evaluated against three baselines. Since the class distribution is unequal, a majority baseline consists in classifying each entity as non-generic. As a second baseline we chose the performance of the feature *Person*, as this feature gave the best performance in precision among those that are similarly easy to extract. Finally, we compare our results to (Suh, 2006).

6 Results and Discussion

The results of classification are summarised in Table 6. The columns Generic and Non-generic give the results for the respective class. Overall shows the weighted average of the classes.

Comparison to baselines Given the bias for non-generic NPs in the unbalanced data, the majority baseline achieves high performance overall (F: 80.6). Of course, it does not detect any generic NPs. The *Person*-based baseline also suffers from very low recall (R: 10.2%), but achieves the highest precision (P: 60.5 %). (Suh, 2006) reported only precision of the generic class, so we can only compare against this value (28.9 %). Most of the features and feature sets yield precision values above the results of Suh.

Feature classes, unbalanced data For the identification of generic NPs, syntactic features achieve the highest precision and recall (P: 40.1%, R: 66.6 %). Using syntactic features on the NP- or sentence-level only, however, leads to a drop in precision as well as recall. The recall achieved by syntactic features can be improved at the cost of precision by adding semantic features (R: 66.6 → 72.1, P: 40.1 → 37). Semantic features in separation perform lower than the syntactic ones, in terms of recall and precision.

Even though our results achieve a lower precision than the *Person* baseline, in terms of f-measure, we achieve a result of over 50%, which is almost three times the baseline.

Feature classes, balanced data Balancing the training data leads to a moderate drop in performance. All feature classes perform lower than on the unbalanced data set, yielding an increase in recall and a drop in precision. The overall performance differences between the balanced and unbalanced data for the best achieved values for the generic class are -4.7 (P), +13.2 (R) and -1.7 (F). This indicates that (i) the features prove to perform rather effectively, and (ii) the distributional bias in the data can be exploited in practical experiments, as long as the data distribution remains constant.

We observe that generally, the recall for the generic class improves for the balanced data. This is most noticeable for the S-level features with an increase of 55 (syntactic) and 29.7 (semantic). This could indicate that S-level features are useful for detecting genericity, but are too sparse in the non-oversampled data to become prominent. This holds especially for the lexical semantic features.

As a general conclusion, syntactic features prove most important in both setups. We also observe that the margin between syntactic and semantic features reduces in the balanced dataset, and that both NP- and S-level features contribute to classification performance, with NP-features generally outperforming the S-level features. This confirms our hypothesis that all feature classes contribute important information.

Feature selection While the above figures were obtained for the entire feature space, we now discuss the effects of feature selection both on performance and the distribution over feature classes. The results for each feature set are given in Table 6. In general, we find a behaviour similar to

	Syntactic	Semantic
NP	Number, Person, Part of Speech, Determiner Type, Bare Plural	Sense[0]
S	Clause.Part of Speech, Dependency Relation[2]	Clause.{Tense, Pred}

Table 7: Best performing features by feature class

the homogeneous classes, in that balanced training data increases recall at the cost of precision.

With respect to overall f-measure, the best single features are strong on the unbalanced data. They even yield a relatively high precision for the generic NPs (49.5%), the highest value among the selected feature sets. This, however, comes at the price of one of the lowest recalls. The best performing feature in terms of f-measure on both balanced and unbalanced data is Set 5 with Set 4 as a close follow-up. Set 5 achieves an f-score of 53.6 (unbalanced) and 51.0 (balanced). The highest recall is achieved using Set 4 (69.6% on the unbalanced and 83.1% on the balanced dataset). The results for Set 5 represent an improvement of 3.5 respectively 2.6 (unbalanced and balanced) over the best achieved results on homogeneous feature classes. In fact, Table 7 shows that these features, selected by ablation testing, distribute over all homogeneous classes.

We trained a decision tree to gain insights into the dependencies among these features. Figure 1 shows an excerpt of the obtained tree. The classifier learned to classify singular proper names as non-generic, while the genericity of singular nouns depends on their predicate. At this point, the classifier can correctly classify some of the NPs in (5) as kind-referring (given the training data contains predicates like "widespread", "die out", ...).

7 Conclusions and Future Work

This paper addresses a linguistic phenomenon that has been thoroughly studied in the formal semantics literature but only recently is starting to be addressed as a task in computational linguistics. We presented a data-driven machine learning approach for identifying generic NPs in context that in turn can be used to improve tasks such as knowledge acquisition and organisation. The classification of generic NPs has proven difficult even for humans. Therefore, a machine learning approach seemed promising, both for the identification of relevant features as for capturing contextual factors. We explored a range of features using homogeneous and mixed classes gained by alternative methods of feature selection. In terms of f-measure on the generic class, all feature sets performed above the baseline(s). In the overall classification, the selected sets perform above the majority and close to or above the *Person* baseline.

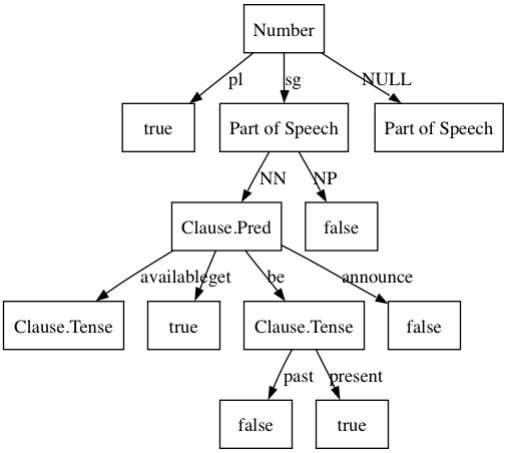


Figure 1: A decision tree trained on feature Set 5

The final feature set that we established characterises generic NPs as a phenomenon that exhibits both syntactic and semantic as well as sentence- and NP-level properties. Although our results are satisfying, in future work we will extend the range of features for further improvements. In particular, we will address lexical semantic features, as they tend to be effected by sparsity. As a next step, we will apply our approach to the classification of generic sentences. Treating both cases simultaneously could reveal insights into dependencies between them.

The classification of generic expressions is only a first step towards a full treatment of the challenges involved in their semantic processing. As discussed, this requires a contextually appropriate selection of the quantifier restriction[8], as well as determining inheritance of properties from classes to individuals and the formalisation of defaults.

References

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. CELEX2. Linguistic Data Consortium, Philadelphia.

Johan Bos. 2009. Applying automated deduction to natural language understanding. *Journal of Applied*

[8]Consider example (1.a), which is contextually restricted to a certain time and space.

Logic, 7(1):100 – 112. Special Issue: Empirically Successful Computerized Reasoning.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Marsuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of Grammar Engineering and Evaluation Workshop*.

Gregory Norman Carlson. 1977. *Reference to Kinds in English*. Ph.D. thesis, University of Massachusetts.

Philipp Cimiano. 2006. *Ontology Learning and Populating from Text*. Springer.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman, 2010. *XLE Documentation*. www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html.

Östen Dahl. 1975. On Generics. In Edward Keenan, editor, *Formal Semantics of Natural Language*, pages 99–111. Cambridge University Press, Cambridge.

Renaat Declerck. 1991. The Origins of Genericity. *Linguistics*, 29:79–102.

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

Lisa Ferro, Laurie Gerber, Janet Hitzeman, Elizabeth Lima, and Beth Sundheim. 2005. ACE English Training Data. Linguistic Data Consortium, Philadelphia.

Michael Gelfond. 2007. Answer sets. In *Handbook of Knowledge Representation*. Elsevier Science.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

Irene Heim. 1982. *The Semantics of Definite and Indefinite Noun Phrases*. Ph.D. thesis, University of Massachusetts, Amherst.

Aurelie Herbelot and Ann Copestake. 2008. Annotating genericity: How do humans decide? (a case study in ontology extraction). In Sam Featherston and Susanne Winkler, editors, *The Fruits of Empirical Linguistics*, volume 1. de Gruyter.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430.

Manfred Krifka, Francis Jeffry Pelletier, Gregory N. Carlson, Alice ter Meulen, Gennaro Chierchia, and Godehard Link. 1995. Genericity: An Introduction. In Gregory Norman Carlson and Francis Jeffry Pelletier, editors, *The Generic Book*. University of Chicago Press, Chicago.

Douglas B. Lenat. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38.

Vladimir Lifschitz. 2002. Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39 – 54.

Vladimir Lifschitz. 2008. What is Answer Set Programming? In *Proceedings of AAAI*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

Thomas Mathew and Graham Katz. 2009. Supervised Categorization of Habitual and Episodic Sentences. In *Sixth Midwest Computational Linguistics Colloquium*. Bloomington, Indiana: Indiana University.

Alexis Mitchell, Stephanie Strassel, Mark Przybocki, JK Davis, George Doddington, Ralph Grishman, Adam Meyers, Ada Brunstein, Lisa Ferro, and Beth Sundheim. 2003. ACE-2 Version 1.0. Linguistic Data Consortium, Philadelphia.

Ian Niles and Adam Pease. 2001. Towards a Standard Upper Ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*.

Athina Pappas and Susan A. Gelman. 1998. Generic noun phrases in mother–child conversations. *Journal of Child Language*, 25(1):19–33.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd Conference on the Advancement of Artificial Intelligence*, pages 1440–1445, Vancouver, B.C., Canada, July.

Willard Van Orman Quine. 1960. *Word and Object*. MIT Press, Cambridge, Massachusetts.

Raymond Reiter. 1980. A logic for default reasoning. *Artificial Intelligence*, 13:81–132.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. *Proceedings of the conference on New Methods in Language Processing*, 12.

Sangweon Suh. 2006. Extracting Generic Statements for the Semantic Web. Master's thesis, University of Edinburgh.

Ian H. Witten and Eibe Frank. 2002. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):76–77.

Cäcilia Zirn, Vivi Nastase, and Michael Strube. 2008. Distinguishing between instances and classes in the Wikipedia taxonomy. In *Proceedings of the 5th European Semantic Web Conference*.

Structural Semantic Relatedness: A Knowledge-Based Method to Named Entity Disambiguation

Xianpei Han **Jun Zhao***

National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
Beijing 100190, China
{xphan,jzhao}@nlpr.ia.ac.cn

Abstract

Name ambiguity problem has raised urgent demands for efficient, high-quality named entity disambiguation methods. In recent years, the increasing availability of large-scale, rich semantic knowledge sources (such as *Wikipedia* and *WordNet*) creates new opportunities to enhance the named entity disambiguation by developing algorithms which can exploit these knowledge sources at best. The problem is that these knowledge sources are heterogeneous and most of the semantic knowledge within them is embedded in complex structures, such as graphs and networks. This paper proposes a knowledge-based method, called *Structural Semantic Relatedness* (*SSR*), which can enhance the named entity disambiguation by capturing and leveraging the structural semantic knowledge in multiple knowledge sources. Empirical results show that, in comparison with the classical BOW based methods and social network based methods, our method can significantly improve the disambiguation performance by respectively 8.7% and 14.7%.

1 Introduction

Name ambiguity problem is common on the Web. For example, the name "Michael Jordan" represents more than ten persons in the Google search results. Some of them are shown below:

Michael (Jeffrey) Jordan, Basketball Player
Michael (I.) Jordan, Professor of Berkeley
Michael (B.) Jordan, American Actor

The name ambiguity has raised serious problems in many relevant areas, such as web person search, data integration, link analysis and knowledge base population. For example, in response to a person query, search engine returns a long, flat list of results containing web pages about several namesakes. The users are then forced either to refine their query by adding terms, or to browse through the search results to find the person they are seeking. Besides, an ever-increasing number of question answering and information extraction systems are coming to rely on data from multi-sources, where name ambiguity will lead to wrong answers and poor results. For example, in order to extract the birth date of the Berkeley professor *Michael Jordan*, a system may return the birth date of his popular namesakes, e.g., the basketball player *Michael Jordan*.

So there is an urgent demand for efficient, high-quality named entity disambiguation methods. Currently, the common methods for named entity disambiguation include name observation clustering (Bagga and Baldwin, 1998) and entity linking with knowledge base (McNamee and Dang, 2009). In this paper, we focus on the method of name observation clustering. Given a set of observations $O = \{o_1, o_2, \ldots, o_n\}$ of the target name to be disambiguated, a named entity disambiguation system should group them into a set of clusters $C = \{c_1, c_2, \ldots, c_m\}$, with each resulting cluster corresponding to one specific entity. For example, consider the following four observations of *Michael Jordan*:

1) ***Michael Jordan*** *is a researcher in Computer Science.*
2) ***Michael Jordan*** *plays basketball in Chicago Bulls.*
3) ***Michael Jordan*** *wins NBA MVP.*
4) *Learning in Graphical Models:* ***Michael Jordan****.*

A named entity disambiguation system should group the 1st and 4th *Michael Jordan* observations into one cluster for they both refer to the Berke-

* Corresponding author

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 50–59,
Uppsala, Sweden, 11-16 July 2010.

ley professor *Michael Jordan*, meanwhile group the other two *Michael Jordan* into another cluster as they refer to another person, the Basketball Player *Michael Jordan*.

To a human, named entity disambiguation is usually not a difficult task as he can make decisions depending on not only contextual clues, but also the prior background knowledge. For example, as shown in Figure 1, with the background knowledge that both *Learning* and *Graphical models* are the topics related to *Machine learning*, while *Machine learning* is the sub domain of *Computer science*, a human can easily determine that the two *Michael Jordan* in the 1st and 4th observations represent the same person. In the same way, a human can also easily identify that the two *Michael Jordan* in the 2nd and 3rd observations represent the same person.

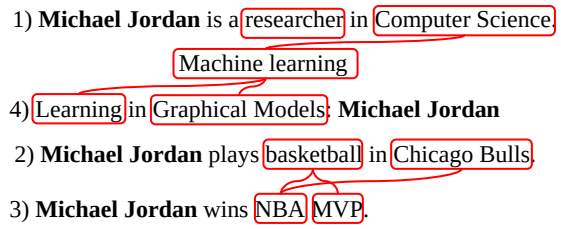


Figure 1. The exploitation of knowledge in human named entity disambiguation

The development of systems which could replicate the human disambiguation ability, however, is not a trivial task because it is difficult to capture and leverage the semantic knowledge as humankind. Conventionally, the named entity disambiguation methods measure the similarity between name observations using the *bag of words* (BOW) model (Bagga and Baldwin (1998); Mann and Yarowsky (2006); Fleischman and Hovy (2004); Pedersen et al. (2005)), where a name observation is represented as a feature vector consisting of the contextual terms. This model measures similarity based on only the co-occurrence statistics of terms, without considering all the semantic relations like social relatedness between named entities, associative relatedness between concepts, and lexical relatedness (e.g., acronyms, synonyms) between key terms.

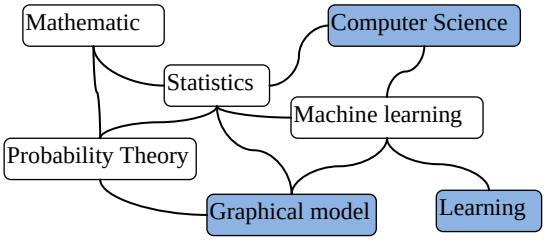


Figure 2. Part of the link structure of Wikipedia

Fortunately, in recent years, due to the evolution of Web (e.g., the *Web 2.0* and the *Semantic Web*) and many research efforts for the construction of knowledge bases, there is an increasing availability of large-scale knowledge sources, such as *Wikipedia* and *WordNet*. These large-scale knowledge sources create new opportunities for knowledge-based named entity disambiguation methods as they contain rich semantic knowledge. For example, as shown in Figure 2, the link structure of Wikipedia contains rich semantic relations between concepts. And we believe that the disambiguation performance can be greatly improved by designing algorithms which can exploit these knowledge sources at best.

The problem of these knowledge sources is that they are heterogeneous (e.g., they contain different types of semantic relations and different types of concepts) and most of the semantic knowledge within them is embedded in complex structures, such as graphs and networks. For example, as shown in Figure 2, the semantic relation between *Graphical Model* and *Computer Science* is embedded in the link structure of the Wikipedia. In recent years, some research has investigated to exploit some specific semantic knowledge, such as the social connection between named entities in the Web (Kalashnikov et al. (2008), Wan et al. (2005) and Lu et al. (2007)), the ontology connection in DBLP (Hassell et al., 2006) and the semantic relations in Wikipedia (Cucerzan (2007), Han and Zhao (2009)). These knowledge-based methods, however, usually are specialized to the knowledge sources they used, so they often have the knowledge coverage problem. Furthermore, these methods can only exploit the semantic knowledge to a limited extent because they cannot take the structural semantic knowledge into consideration.

To overcome the deficiencies of previous methods, this paper proposes a knowledge-based method, called *Structural Semantic Relatedness* (*SSR*), which can enhance the named entity disambiguation by capturing and leveraging the structural semantic knowledge from multiple knowledge sources. The key point of our method is a reliable semantic relatedness measure between concepts (including WordNet concepts, NEs and Wikipedia concepts), called *Structural Semantic Relatedness*, which can capture both the explicit semantic relations between concepts and the implicit semantic knowledge embedded in graphs and networks. In particular, we first extract the semantic relations between two concepts from a variety of knowledge sources and

represent them using a graph-based model, *semantic-graph*. Then based on the principle that "two concepts are semantic related if they are both semantic related to the neighbor concepts of each other", we construct our *Structural Semantic Relatedness* measure. In the end, we leverage the structural semantic relatedness measure for named entity disambiguation and evaluate the performance on the standard WePS data sets. The experimental results show that our *SSR* method can significantly outperform the traditional methods.

This paper is organized as follows. Section 2 describes how to construct the structural semantic relatedness measure. Next in Section 3 we describe how to leverage the captured knowledge for named entity disambiguation. Experimental results are demonstrated in Sections 4. Section 5 briefly reviews the related work. Section 6 concludes this paper and discusses the future work.

2 The Structural Semantic Relatedness Measure

In this section, we demonstrate the structural semantic relatedness measure, which can capture the structural semantic knowledge in multiple knowledge sources. Totally, there are two problems we need to address:

1) How to extract and represent the semantic relations between concepts, since there are many types of semantic relations and they may exist as different patterns (the semantic knowledge may exist as explicit semantic relations or be embedded in complex structures).

2) How to capture all the extracted semantic relations between concepts in our semantic relatedness measure.

To address the above two problems, in following we first introduce how to extract the semantic relations from multiple knowledge sources; then we represent the extracted semantic relations using the semantic-graph model; finally we build our structural semantic relatedness measure.

2.1 Knowledge Sources

We extract three types of semantic relations (semantic relatedness between Wikipedia concepts, lexical relatedness between WordNet concepts and social relatedness between NEs) correspondingly from three knowledge sources: Wikipedia, WordNet and NE Co-occurrence Corpus.

1. **Wikipedia**[1], a large-scale online encyclopedia, its English version includes more than 3,000,000 concepts and new articles are added quickly and up-to-date. Wikipedia contains rich semantic knowledge in the form of hyperlinks between Wikipedia articles, such as *Polysemy* (disambiguation pages), *Synonym* (redirect pages) and *Associative relation* (hyperlinks between Wikipedia articles). In this paper, we extract the semantic relatedness *sr* between Wikipedia concepts using the method described in Milne and Witten(2008):

$$sr(a,b) = 1 - \frac{\log(\max(|A|,|B|)) - \log(|A \cap B|)}{\log(|W|) - \log(\min(|A|,|B|))}$$

where *a* and *b* are the two concepts of interest, *A* and *B* are the sets of all the concepts that are respectively linked to *a* and *b*, and *W* is the entire Wikipedia. For demonstration, we show the semantic relatedness between four selected concepts in Table 1.

	Statistics	**Basketball**
Machine learning	0.58	0.00
MVP	0.00	0.45

Table 1. The semantic relatedness table of four selected Wikipedia concepts

2. **WordNet 3.0**[2] (Fellbaum et al., 1998), a lexical knowledge source includes over 110,000 WordNet concepts (word senses about English words). Various lexical relations are recorded between WordNet concepts, such as *hyponyms*, *holonym* and *synonym*. The lexical relatedness *lr* between two WordNet concepts are measured using the Lin (1998)'s WordNet semantic similarity measure. Table 2 shows some examples of the lexical relatedness.

	school	**science**
university	0.67	0.10
research	0.54	0.39

Table 2. The lexical relatedness table of four selected WordNet concepts

3. **NE Co-occurrence Corpus**, a corpus of documents for capturing the social relatedness between named entities. According to the fuzzy set theory (Baeza-Yates et al., 1999), the degree of named entities co-occurrence in a corpus is a measure of the relatedness between them. For example, in Google search results, the "Chicago Bulls" co-occurs with "NBA" in more than

[1] http://www.wikipedia.org/
[2] http:// wordnet.princeton.edu/

7,900,000 web pages, while only co-occurs with "EMNLP" in less than *1,000* web pages. So the co-occurrence statistics can be used to measure the social relatedness between named entities. In this paper, given a NE Co-occurrence Corpus *D*, the social relatedness *scr* between two named entities ne_1 and ne_2 is measured using the Google Similarity Distance (Cilibrasi and Vitanyi, 2007):

$$scr(ne_1, ne_2) = 1 - \frac{\log(\max(|D_1|, |D_2|)) - \log(|D_1 \cap D_2|)}{\log(|D|) - \log(\min(|D_1|, |D_2|))}$$

where D_1 and D_2 are the document sets correspondingly containing ne_1 and ne_2. An example of social relatedness is shown in Table 3, which is computed using the Web corpus through Google.

	ACL	NBA
EMNLP	0.61	0.00
Chicago Bulls	0.19	0.55

Table 3. The social relatedness table of four selected named entities

2.2 The Semantic-Graph Model

In this section we present a graph-based representation, called *semantic-graph*, to model the extracted semantic relations as a graph within which the semantic relations are interconnected and transitive. Concretely, the semantic-graph is defined as follows:

*A **semantic-graph** is a weighted graph **G = (V, E)**, where each node represents a distinct concept; and each edge between a pair of nodes represents the semantic relation between the two concepts corresponding to these nodes, with the edge weight indicating the strength of the semantic relation.*

For demonstration, Figure 3 shows a semantic-graph which models the semantic knowledge extracted from Wikipedia for the *Michael Jordan* observations in Section 1.

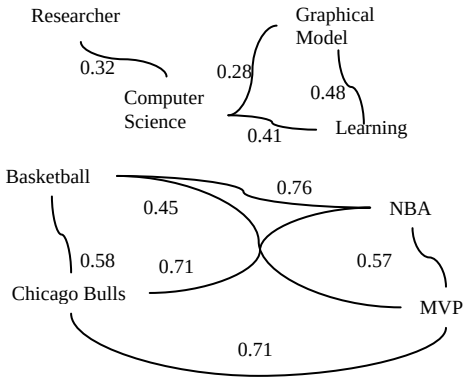


Figure 3. An example of semantic-graph

Given a set of name observations, the construction of semantic-graph takes two steps: concept extraction and concept connection. In the following we respectively describe each step.

1) Concept Extraction. In this step we extract all the concepts in the contexts of name observations and represent them as the nodes in the semantic-graph. We first gather all the N-grams (up to 8 words) and identify whether they correspond to semantically meaningful concepts: if a N-gram is contained in the WordNet, we identify it as a WordNet concept, and use its primary word sense as its semantic meaning; to find whether a N-gram is a named entity, we match it to the named entity list extracted using the open-Calais API3, which contains more than 30 types of named entities, such as Person, Organization and Award; to find whether a N-gram is a Wikipedia concept, we match it to the Wikipedia anchor dictionary, then find its corresponding Wikipedia concept using the method described in (Medelyan et al, 2008). After concept identification, we filter out all the N-grams which do not correspond to the semantic meaningful concepts, such as the N-grams "*learning in*" and "*wins NBA MVP*". The retained N-grams are identified as concepts, corresponding with their semantic meanings (a concept may have multiple semantic meaning explanation, e.g., the "*MVP*" has three semantic meaning, as "*most valuable player, MVP*" in WordNet, as the "*Most Valuable Player*" in Wikipedia and as a named entity of Award type).

2) Concept Connection. In this step we represent the semantic relations as the edges between nodes. That is, for each pair of extracted concepts, we identify whether there are semantic relations between them: 1) If there is only one semantic relation between them, we connect these two concepts with an edge, where the edge weight is the strength of the semantic relation; 2) If there is more than one semantic relations between them, we choose the most reliable semantic relation, i.e., we choose the semantic relation in the knowledge sources according to the order of WordNet, Wikipedia and NE Co-concurrence corpus (Suchanek et al., 2007). For example, if both Wikipedia and WordNet provide the semantic relation between *MVP* and *NBA*, we choose the semantic relation provided by WordNet.

[3] http://www.opencalais.com/

2.3 The Structural Semantic Relatedness Measure

In this section, we describe how to capture the semantic relations between the concepts in semantic-graph using a semantic relatedness measure. Totally, the semantic knowledge between concepts is modeled in two forms:

1) The edges of semantic-graph. The edges model the direct semantic relations between concepts. We call this form of semantic knowledge as *explicit semantic knowledge*.

2) The structure of semantic-graph. Except for the edges, the structure of the semantic-graph also models the semantic knowledge of concepts. For example, the neighbors of a concept represent all the concepts which are explicitly semantic-related to this concept; and the paths between two concepts represent all the explicit and implicit semantic relations between them. We call this form of semantic knowledge as *structural semantic knowledge*, or *implicit semantic knowledge*.

Therefore, in order to deduce a reliable semantic relatedness measure, we must take both the edges and the structure of semantic-graph into consideration. Under the semantic-graph model, the measurement of semantic relatedness between concepts equals to quantifying the similarity between nodes in a weighted graph. To simplify the description, we assign each node in semantic-graph an integer index from 1 to $|\boldsymbol{V}|$ and use this index to represent the node, then we can write the adjacency matrix of the semantic-graph $\boldsymbol{G}$ as $\boldsymbol{A}$, where $\boldsymbol{A[i,j]}$ or $\boldsymbol{A_{ij}}$ is the edge weight between node *i* and node *j*.

The problem of quantifying the relatedness between nodes in a graph is not a new problem, e.g., the *structural equivalence* and *structural similarity* (the SimRank in Jeh and Widom (2002) and the similarity measure in Leicht et al. (2006)). However, these similarity measures are not suitable for our task, because all of them assume that the edges are uniform so that they cannot take edge weight into consideration.

In order to take both the graph structure and the edge weight into account, we design the structural semantic relatedness measure by extending the measure introduced in Leicht et al. (2006). The fundamental principle behind our measure is "a node *u* is semantically related to another node *v* if its immediate neighbors are semantically related to *v*". This definition is natural, for example, as shown in Figure 3, the concept *Basketball* and its neighbors *NBA* and *Chicago Bulls* are all semantically related to *MVP*. This definition is recursive, and the starting point we choose is the semantic relatedness in the edge. Thus our structural semantic relatedness has two components: the neighbor term of the previous recursive phase which captures the graph structure and the semantic relatedness which captures the edge information. Thus, the recursive form of the structural semantic relatedness S_{ij} between the node *i* and the node *j* can be written as:

$$S_{ij} = \lambda \sum_{l \in N_i} \frac{A_{il}}{d_i} S_{lj} + \mu A_{ij}$$

where λ and μ control the relative importance of the two components and

$N_i=\{j \mid A_{ij} > 0\}$ is the set of the immediate neighbors of node *i*;

$d_i = \sum_{j \in N_i} A_{ij}$ is the degree of node *i*.

In order to solve this formula, we introduce the following two notations:

$\boldsymbol{T}$: The relatedness transition matrix, where $T[i,j]=A_{ij}/d_i$, indicating the transition rate of relatedness from node *j* to its neighbor *i*.

$\boldsymbol{S}$: The structural semantic relatedness matrix, where $S[i,j]=S_{ij}$.

Now we can turn our first form of structural semantic relatedness into the matrix form:

$$S = \lambda TS + \mu A$$

By solving this equation, we can get:

$$S = \mu(I - \lambda T)^{-1} A$$

where $\boldsymbol{I}$ is the identity matrix. Since μ is a parameter which only contributes an overall scale factor to the relatedness value, we can ignore it and get the final form of the structural semantic relatedness as:

$$S = (I - \lambda T)^{-1} A$$

Because the *S* is asymmetric, the finally relatedness between node *i* and node *j* is the average of S_{ij} and S_{ji}.

The meaning of λ : The last question of our structural semantic relatedness measure is how to set the free parameter λ . To understand the meaning of λ**,** let us expand the similarity as a power series thus:

$$S = (I + \lambda T + \lambda^2 T^2 + ... + \lambda^k T^k + ...)A$$

Noting that the $[T^k]_{ij}$ element is the relatedness transition rate from node *i* to node *j* with path length *k*, we can view the λ as a penalty factor for the transition path length: by setting the λ with a value within (0, 1), a longer graph path will contribute less to the final relatedness value. The optimal value of λ is 0.6 through a learning

process shown in Section 4. For demonstration, Table 4 shows some structural semantic relatedness values of the Semantic-graph in Figure 3 (*CS* represents *computer science* and *GM* represents *Graphical model*). From Table 4, we can see that the structural semantic relatedness can successfully capture the semantic knowledge embedded in the structure of semantic-graph, such as the implicit semantic relation between *Researcher* and *Learning*.

	Researcher	CS	GM	Learning
Researcher	---	0.50	0.27	0.31
CS	0.50	---	0.62	0.73
GM	0.27	0.62	---	0.80
Learning	0.31	0.73	0.80	---

Table 4. The structural semantic relatedness of the semantic-graph shown in Figure 3

3 Named Entity Disambiguation by Leveraging Semantic Knowledge

In this section we describe how to leverage the semantic knowledge captured in the structural semantic relatedness measure for named entity disambiguation. Because the key problem of named entity disambiguation is to measure the similarity between name observations, we integrate the structural semantic relatedness in the similarity measure, so that it can better reflect the actual similarity between name observations.

Concretely, our named entity disambiguation system works as follows: 1) Measuring the similarity between name observations; 2) Grouping name observations using the clustering algorithm. In the following we describe each step in detail.

3.1 Measuring the Similarity between Name Observations

Intuitively, if two observations of the target name represent the same entity, it is highly possible that the concepts in their contexts are closely related, i.e., the named entities in their contexts are socially related and the Wikipedia concepts in their contexts are semantically related. In contrast, if two name observations represent different entities, the concepts within their contexts will not be closely related. Therefore we can measure the similarity between two name observations by summarizing all the semantic relatedness between the concepts in their contexts.

To measure the similarity between name observations, we represent each name observation as a weighted vector of concepts (including named entities, Wikipedia concepts and WordNet concepts), where the concepts are extracted using the same method described in Section 2.2, so they are just the same concepts within the semantic-graph. Using the same concept index as the semantic-graph, a name observation o_i is then represented as $o_i = \{w_{i1}, w_{i2}, ..., w_{in}\}$, where w_{ik} is the k^{th} concept's weight in observation o_i, computed using the standard TFIDF weight model, where the DF is computed using the Google Web1T 5-gram corpus[4]. Given the concept vector representation of two name observations o_i and o_j, their similarity is computed as:

$$SIM(o_i, o_j) = \sum_l \sum_k w_{il} w_{jk} S_{lk} \Big/ \sum_l \sum_k w_{il} w_{jk}$$

which is the weighted average of all the structural semantic relatedness between the concepts in the contexts of the two name observations.

3.2 Grouping Name Observations through Hierarchical Agglomerative Clustering

Given the computed similarities, name observations are disambiguated by grouping them according to their represented entities. In this paper, we group name observations using the hierarchical agglomerative clustering(HAC) algorithm, which is widely used in prior disambiguation research and evaluation task (WePS1 and WePS2). The HAC produce clusters in a bottom-up way as follows: Initially, each name observation is an individual cluster; then we iteratively merge the two clusters with the largest similarity value to form a new cluster until this similarity value is smaller than a preset merging threshold or all the observations reside in one common cluster. The merging threshold can be determined through cross-validation. We employ the single-link method to compute the similarity between two clusters, which has been applied widely in prior research (Bagga and Baldwin (1998); Mann and Yarowsky (2003)).

4 Experiments

To assess the performance of our method and compare it with traditional methods, we conduct a series of experiments. In the experiments, we evaluate the proposed *SSR* method on the task of personal name disambiguation, which is the most common type of named entity disambiguation. In the following, we first explain the general experimental settings in Section 4.1, 4.2 and 4.3; then evaluate and discuss the performance of our method in Section 4.4.

[4] www.ldc.upenn.edu/Catalog/docs/LDC2006T13/

4.1 Disambiguation Data Sets

We adopted the standard data sets used in the First Web People Search Clustering Task (**WePS1**) (Artiles et al., 2007) and the Second Web People Search Clustering Task (**WePS2**) (Artiles et al., 2009). The three data sets we used are **WePS1_training** data set, **WePS1_test** data set, and **WePS2_test** data set. Each of the three data sets consists of a set of ambiguous personal names (totally 109 personal names); and for each name, we need to disambiguate its observations in the web pages of the top *N* (100 for **WePS1** and 150 for **WePS2**) Yahoo! search results.

The experiment made the standard "one person per document" assumption, which is widely used in the participated systems in WePS1 and WePS2, i.e., all the observations of the same name in a document are assumed to represent the same entity. Based on this assumption, the features within the entire web page are used to disambiguate personal names.

4.2 Knowledge Sources

There were three knowledge sources we used for our experiments: the WordNet 3.0; the Sep. 9, 2007 English version of Wikipedia; and the Web pages of each ambiguous name in WePS datasets as the NE Co-occurrence Corpus.

4.3 Evaluation Criteria

We adopted the measures used in WePS1 to evaluate the performance of name disambiguation. These measures are:

Purity (**Pur**): *measures the homogeneity of name observations in the same cluster*;

Inverse purity (**Inv_Pur**): *measures the completeness of a cluster*;

F-Measure (**F**): *the harmonic mean of purity and inverse purity*.

The detailed definitions of these measures can be found in Amigo, et al. (2008). We use F-measure as the primary measure just liking WePS1 and WePS2.

4.4 Experimental Results

We compared our method with four baselines: (1) ***BOW:*** The first one is the traditional *Bag of Words* model (*BOW*) based methods: hierarchical agglomerative clustering (HAC) over term vector similarity, where the features including single words and NEs, and all the features are weighted using TFIDF. This baseline is also the state-of-art method in WePS1 and WePS2. (2) ***SocialNetwork:*** The second one is the social network based methods, which is the same as the method described in Malin et al. (2005): HAC over the similarity obtained through random walk over the social network built from the web pages of the top *N* search results. (3)***SSR-NoKnowledge:*** The third one is used as a baseline for evaluating the efficiency of semantic knowledge: HAC over the similarity computed on semantic-graph with no knowledge integrated, i.e., the similarity is computed as:

$$SIM(o_i, o_j) = \sum_l w_{il} w_{jl} \Big/ \sum_l \sum_k w_{il} w_{jk}$$

(4) ***SSR-NoStructure***: The fourth one is used as a baseline for evaluating the efficiency of the semantic knowledge embedded in complex structures: HAC over the similarity computed by only integrating the explicit semantic relations, i.e., the similarity is computed as:

$$SIM(o_i, o_j) = \sum_l \sum_k w_{il} w_{jk} A_k \Big/ \sum_l \sum_k w_{il} w_{jk}$$

4.4.1 Overall Performance

We conducted several experiments on all the three WePS data sets: the four baselines, the proposed ***SSR*** method and the proposed ***SSR*** method with only one special type knowledge added, respectively ***SSR-NE***, ***SSR-WordNet*** and ***SSR-Wikipedia***. All the optimal merging thresholds used in HAC were selected by applying leave-one-out cross validation. The overall performance is shown in Table 5.

Method	WePS1_training		
	Pur	**Inv_Pur**	**F**
BOW	0.71	0.88	**0.78**
SocialNetwork	0.66	0.98	**0.76**
SSR-NoKnowledge	0.79	0.89	**0.81**
SSR-NoStructure	0.87	0.83	**0.83**
SSR-NE	0.80	0.86	**0.82**
SSR-WordNet	0.80	0.91	**0.83**
SSR-Wikipedia	0.82	0.90	**0.84**
SSR	0.82	0.92	**0.85**
	WePS1_test		
	Pur	**Inv_Pur**	**F**
BOW	0.74	0.87	**0.74**
SocialNetwork	0.83	0.63	**0.65**
SSR-NoKnowledge	0.80	0.74	**0.75**
SSR-NoStructure	0.80	0.78	**0.78**
SSR-NE	0.73	0.80	**0.74**
SSR-WordNet	0.81	0.77	**0.77**
SSR-Wikipedia	0.88	0.77	**0.81**
SSR	0.85	0.83	**0.84**
	WePS2_test		
	Pur	**Inv_Pur**	**F**
BOW	0.80	0.80	**0.77**
SocialNetwork	0.62	0.93	**0.70**
SSR-NoKnowledge	0.84	0.80	**0.80**
SSR-NoStructure	0.84	0.83	**0.81**
SSR-NE	0.78	0.88	**0.80**
SSR-WordNet	0.85	0.82	**0.83**
SSR-Wikipedia	0.84	0.81	**0.82**
SSR	0.89	0.84	**0.86**

Table 5. Performance results of baselines and SSR methods

From the performance results in Table 5, we can see that:

1) The semantic knowledge can greatly improve the disambiguation performance: compared with the BOW and the SocialNetwork baselines, SSR respectively gets 8.7% and 14.7% improvement on average on the three data sets.

2) By leveraging the semantic knowledge from multiple knowledge sources, we can obtain a better named entity disambiguation performance: compared with the *SSR-NE*'s 0% improvement, the *SSR-WordNet*'s 2.3% improvement and the *SSR-Wikipedia*'s 3.7% improvement, the *SSR* gets 6.3% improvement over the *SSR-NoKnowledge* baseline, which is larger than all the *SSR* methods with only one type of semantic knowledge integrated.

3) The exploitation of the structural semantic knowledge can further improve the disambiguation performance: compared with *SSR-NoStructure*, our *SSR* method achieves 4.3% improvement.

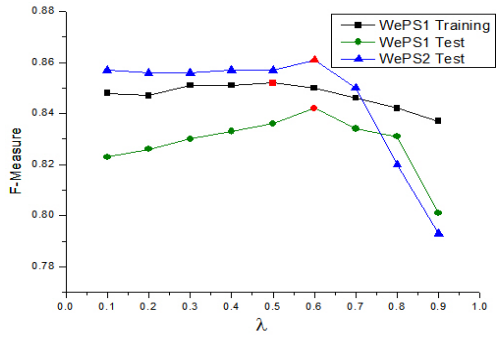


Figure 4. The F-Measure vs. λ on three data sets

4.4.2 *Optimizing Parameters*

There is only one parameter λ needed to be configured, which is the penalty factor for the relatedness transition path length in the structural semantic relatedness measure. Usually a smaller λ will make the structural semantic knowledge contribute less in the resulting relatedness value. Figure 4 plots the performance of our method corresponding to the special λ settings. As shown in Figure 4, the SSR method is not very sensitive to the λ and can achieve its best average performance when the value of λ is *0.6*.

4.4.3 *Detailed Analysis*

To better understand the reasons why our *SSR* method works well and how the exploitation of structural semantic knowledge can improve performance, we analyze the results in detail.

The Exploitation of Semantic Knowledge. The primary advantage of our method is the exploitation of semantic knowledge. Our method exploits the semantic knowledge in two directions:

1) *The Integration of Multiple Semantic Knowledge Sources.* Using the semantic-graph model, our method can integrate the semantic knowledge extracted from multiple knowledge sources, while most traditional knowledge-based methods are usually specialized to one type of knowledge. By integrating multiple semantic knowledge sources, our method can improve the semantic knowledge coverage.

2) *The exploitation of Semantic Knowledge embedded in complex structures.* Using the structural semantic relatedness measure, our method can exploit the implicit semantic knowledge embedded in complex structures; while traditional knowledge-based methods usually lack this ability.

The Rich Meaningful Features. One another advantage of our method is the rich meaningful features, which is brought by the multiple semantic knowledge sources. With more meaningful features, our method can better describe the name observations with less information loss. Furthermore, unlike the traditional N-gram features, the features enriched by semantic knowledge sources are all semantically meaningful units themselves, so little noisy features will be added. The effect of rich meaningful features can also be shown in Table 5: by adding these features, the *SSR-NoKnowledge* respectively achieves 2.3% and 9.7% improvement over the *BOW* and the *SocialNetwork* baseline.

5 Related Work

In this section, we briefly review the related work. Totally, the traditional named entity disambiguation methods can be classified into two categories: the shallow methods and the knowledge-based methods.

Most of previous named entity disambiguation researches adopt the shallow methods, which are mostly the natural extension of the *bag of words* (*BOW*) model. Bagga and Baldwin (1998) represented a name as a vector of its contextual words, then two names were predicted to be the same entity if their cosine similarity is above a threshold. Mann and Yarowsky (2003) and Niu et al. (2004) extended the vector representation with extracted biographic facts. Pedersen et al. (2005) employed significant bigrams to represent

a name observation. Chen and Martin (2007) explored a range of syntactic and semantic features.

In recent years some research has investigated employing knowledge sources to enhance the named entity disambiguation. Bunescu and Pasca (2006) disambiguated the names using the category information in Wikipedia. Cucerzan (2007) disambiguated the names by combining the *BOW* model with the Wikipedia category information. Han and Zhao (2009) leveraged the Wikipedia semantic knowledge for computing the similarity between name observations. Bekkerman and McCallum (2005) disambiguated names based on the link structure of the Web pages between a set of socially related persons. Kalashnikov et al. (2008) and Lu et al. (2007) used the co-occurrence statistics between named entities in the Web. The social network was also exploited for named entity disambiguation, where similarity is computed through random walking, such as the work introduced in Malin (2005), Malin and Airoldi (2005), Yang et al.(2006) and Minkov et al. (2006). Hassell et al. (2006) used the relationships from DBLP to disambiguate names in research domain.

6 Conclusions and Future Works

In this paper we demonstrate how to enhance the named entity disambiguation by capturing and exploiting the semantic knowledge existed in multiple knowledge sources. In particular, we propose a semantic relatedness measure, *Structural Semantic Relatedness*, which can capture both the explicit semantic relations and the implicit structural semantic knowledge. The experimental results on the WePS data sets demonstrate the efficiency of the proposed method. For future work, we want to develop a framework which can uniformly model the semantic knowledge and the contextual clues for named entity disambiguation.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grants no. 60875041 and 60673042, and the National High Technology Development 863 Program of China under Grants no. 2006AA01Z144.

References

Amigo, E., Gonzalo, J., Artiles, J. and Verdejo, F. 2008. A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information Retrieval.

Artiles, J., Gonzalo, J. & Sekine, S. 2007. The SemEval-2007 WePS Evaluation: Establishing a benchmark for the Web People Search Task. In SemEval.

Artiles, J., Gonzalo, J. and Sekine, S. 2009. WePS2 Evaluation Campaign: Overview of the Web People Search Clustering Task. In WePS2, WWW 2009.

Baeza-Yates, R., Ribeiro-Neto, B., et al. 1999. Modern information retrieval. Addison-Wesley Reading, MA.

Bagga, A. & Baldwin, B. 1998. Entity-based cross-document coreferencing using the vector space model. Proceedings of the 17th international conference on Computational linguistics-Volume 1, pp. 79-85.

Bekkerman, R. & McCallum, A. 2005. Disambiguating web appearances of people in a social network. Proceedings of the 14th international conference on World Wide Web, pp. 463-470.

Bunescu, R. & Pasca, M. 2006. Using encyclopedic knowledge for named entity disambiguation. Proceedings of EACL, vol. 6.

Chen, Y. & Martin, J. 2007. Towards robust unsupervised personal name disambiguation. Proceedings of EMNLP and CoNLL, pp. 190-198.

Cilibrasi, R. L., Vitanyi, P. M. & CWI, A. 2007. The google similarity distance, IEEE Transactions on knowledge and data engineering, vol. 19, no. 3, pp. 370-383.

Cucerzan, S. 2007, Large-scale named entity disambiguation based on Wikipedia data. Proceedings of EMNLP-CoNLL, pp. 708-716.

Fellbaum, C., et al. 1998. WordNet: An electronic lexical database. MIT press Cambridge, MA.

Fleischman, M. B. & Hovy, E. 2004. Multi-document person name resolution. Proceedings of ACL, Reference Resolution Workshop.

Han, X. & Zhao, J. 2009. Named entity disambiguation by leveraging Wikipedia semantic knowledge. Proceeding of the 18th ACM conference on Information and knowledge management, pp. 215-224.

Hassell, J., Aleman-Meza, B. & Arpinar, I. 2006. Ontology-driven automatic entity disambiguation in unstructured text. Proceedings of The 2006 ISWC, pp. 44-57.

Jeh, G. & Widom, J. 2002. SimRank: A measure of structural-context similarity, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, p. 543.

Kalashnikov, D. V., Nuray-Turan, R. & Mehrotra, S. 2008. Towards Breaking the Quality Curse. A Web-Querying Approach to Web People Search. In Proc. of SIGIR.

Leicht, E. A., Petter Holme, & M. E. J. Newman. 2006. Vertex similarity in networks. Physical Review E , vol. 73, no. 2, p. 26120.

Lin., D. 1998. An information-theoretic definition of similarity. In Proc. of ICML.

Lu, Y. & Nie , Z. et al. 2007. Name Disambiguation Using Web Connection. In Proc. of AAAI.

Malin, B. 2005. Unsupervised name disambiguation via social network similarity. SIAM SDM Workshop on Link Analysis, Counterterrorism and Security.

Malin, B., Airoldi, E. & Carley, K. M. 2005. A network analysis model for disambiguation of names in lists. Computational & Mathematical Organization Theory, vol. 11, no. 2, pp. 119-139.

Mann, G. S. & Yarowsky, D. 2003. Unsupervised personal name disambiguation, Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, p. 40.

McNamee, P. & Dang, H. Overview of the TAC 2009 Knowledge Base Population Track. In Proceedings of Text Analysis Conference (TAC-2009), 2009.

Medelyan, O., Witten, I. H. & Milne, D. 2008. Topic indexing with Wikipedia. Proceedings of the AAAI WikiAI workshop.

Milne, D., Medelyan, O. & Witten, I. H. 2006. Mining domain-specific thesauri from wikipedia: A case study. IEEE/WIC/ACM International Conference on Web Intelligence, pp. 442-448.

Minkov, E., Cohen, W. W. & Ng, A. Y. 2006. Contextual search and name disambiguation in email using graphs, Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 27-34.

Niu C., Li W. and Srihari, R. K. 2004. Weakly Supervised Learning for Cross-document Person Name Disambiguation Supported by Information Extraction. Proceedings of ACL, pp. 598-605.

Pedersen, T., Purandare, A. & Kulkarni, A. 2005. Name discrimination by clustering similar contexts. Computational Linguistics and Intelligent Text Processing, pp. 226-237.

Strube, M. & Ponzetto, S. P. 2006. WikiRelate! Computing semantic relatedness using Wikipedia, Proceedings of the National Conference on Artificial Intelligence, vol. 21, no. 2, p. 1419.

Suchanek, F. M., Kasneci, G. & Weikum, G. 2007. Yago: a core of semantic knowledge, Proceedings of the 16th international conference on World Wide Web, p. 706.

Wan, X., Gao, J., Li, M. & Ding, B. 2005. Person resolution in person search results: Webhawk. Proceedings of the 14th ACM international conference on Information and knowledge management, p. 170.

Witten, D. M. & Milne, D. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA, pp. 25-30.

Yang, K. H., Chiou, K. Y., Lee, H. M. & Ho, J. M. 2006. Web appearance disambiguation of personal names based on network motif. Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 386-389.

Correcting errors in speech recognition with articulatory dynamics

Frank Rudzicz
University of Toronto, Department of Computer Science
Toronto, Ontario, Canada
frank@cs.toronto.edu

Abstract

We introduce a novel mechanism for incorporating articulatory dynamics into speech recognition with the theory of task dynamics. This system reranks sentence-level hypotheses by the likelihoods of their hypothetical articulatory realizations which are derived from relationships learned with aligned acoustic/articulatory data. Experiments compare this with two baseline systems, namely an acoustic hidden Markov model and a dynamic Bayes network augmented with discretized representations of the vocal tract. Our system based on task dynamics reduces word-error rates significantly by 10.2% relative to the best baseline models.

1 Introduction

Although modern automatic speech recognition (ASR) takes several cues from the biological perception of speech, it rarely models its biological production. The result is that speech is treated as a surface acoustic phenomenon with lexical or phonetic hidden dynamics but without any physical constraints in between. This omission leads to some untenable assumptions. For example, speech is often treated out of convenience as a sequence of discrete, non-overlapping packets, such as phonemes, despite the fact that some major difficulties in ASR, such as co-articulation, are by definition the result of concurrent physiological phenomena (Hardcastle and Hewlett, 1999).

Many acoustic ambiguities can be resolved with knowledge of the vocal tract's configuration (O'Shaughnessy, 2000). For example, the three nasal sonorants, */m/*, */n/*, and */ng/*, are acoustically similar (i.e., they have large concentrations of energy at the same frequencies) but uniquely and reliably involve bilabial closure, tongue-tip elevation, and tongue-dorsum elevation, respectively. Having access to the articulatory goals of the speaker would, in theory, make the identification of linguistic intent almost trivial. Although we don't typically have access to the vocal tract during speech recognition, its configuration *can* be estimated reasonably well from acoustics alone within adequate models or measurements of the vocal tract (Richmond et al., 2003; Toda et al., 2008). Evidence that such inversion takes place naturally in humans during speech perception suggests that the discriminability of speech sounds depends powerfully on their production (Liberman and Mattingly, 1985; D'Ausilio et al., 2009).

This paper describes the use of explicit models of physical speech production within recognition systems. Initially, we augment traditional models of ASR with probabilistic relationships between acoustics and articulation learned from appropriate data. This leads to the incorporation of a high-level, goal-oriented, and control-based theory of speech production within a novel ASR system.

2 Background and related work

The use of theoretical (phonological) features of the vocal tract has provided some improvement over traditional acoustic ASR systems in phoneme recognition with neural networks (Kirchhoff, 1999; Roweis, 1999), but there has been very little work in ASR informed by direct measurements of the vocal tract. Recently, Markov et al. (2006) have augmented hidden Markov models with Bayes networks trained to describe articulatory constraints from a small amount of Japanese vocal tract data, resulting in a small phoneme-error reduction. This work has since been expanded upon to inform ASR systems sensitive to physiological speech disorders (Rudzicz, 2009). Common among previous efforts is an interpretation of speech as a sequence of short, instantaneous observations devoid of long-term dynamics.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 60–68,
Uppsala, Sweden, 11-16 July 2010.

2.1 Articulatory phonology

Articulatory phonology bridges the divide between the physical manifestation of speech and its underlying lexical intentions. Within this discipline, the theory of *task dynamics* is a combined model of physical articulator motion and the planning of abstract vocal tract configurations (Saltzman, 1986). This theory introduces the notion that all observed patterns of speech are the result of overlapping *gestures*, which are abstracted goal-oriented reconfigurations of the vocal tract, such as bilabial closure or velar opening (Saltzman and Munhall, 1989). Each gesture occurs within one of the following *tract variables* (TVs): velar opening (**VEL**), lip aperture (**LA**) and protrusion (**LP**), tongue tip constriction location (**TTCL**) and degree (**TTCD**) [1], tongue body constriction location (**TBCL**) and degree (**TBCD**), lower tooth height (**LTH**), and glottal vibration (**GLO**). For example, the syllable *pub* consists of an onset (*/p/*), a nucleus (*/ah/*), and a coda (*/b/*). Four gestural goals are associated with the onset, namely the shutting of GLO and of VEL, and the closure and release of LA. Similarly, the nucleus of the syllable consists of three goals, namely the relocation of TBCD and TBCL, and the opening of GLO. The presence and extent of these gestural goals are represented by filled rectangles in figure 1. Inter-gestural timings between these goals are specified relative to one another according to human data as described by Nam and Saltzman (2003).

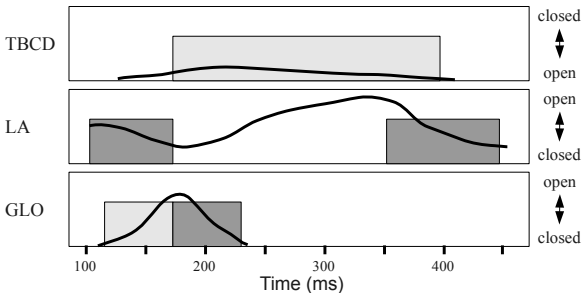


Figure 1: Canonical example *pub* from Saltzman and Munhall (1989).

The presence of these discrete goals influences the vocal tract dynamically and continuously as modelled by the following non-homogeneous second-order linear differential equation:

$$Mz'' + Bz' + K(z - z^*) = 0. \quad (1)$$

[1] Constriction *locations* generally refer to the front-back dimension of the vocal tract and constriction *degrees* generally refer to the top-down dimension.

Here, z is a continuous vector representing the instantaneous positions of the nine tract variables, z^* is the target (equilibrium) positions of those variables, and vectors z' and z'' represent the first and second derivatives of z with respect to time (i.e., velocity and acceleration), respectively. The matrices M, B, and K are syllable-specific coefficients describing the inertia, damping, and stiffness, respectively, of the virtual gestures. Generally, this theory assumes that the tract variables are mutually independent, and that the system is critically damped (i.e., the tract variables do not oscillate around their equilibrium positions) (Nam and Saltzman, 2003). The continuous state, z, of equation (1) is exemplified by black curves in figure 1.

2.2 Articulatory data

Tract variables provide the dimensions of an abstract gestural space independent of the physical characteristics of the speaker. In order to complete our articulatory model, however, we require physical data from which to infer these high-level articulatory goals.

Electromagnetic articulography (EMA) is a method to measure the motion of the vocal tract during speech. In EMA, the speaker is placed within a low-amplitude electromagnetic field produced within a cube of a known geometry. Tiny sensors within this field induce small electric currents whose energy allows the inference of articulator positions and velocities to within 1 mm of error (Yunusova et al., 2009). We derive data for the following study from two EMA sources:

- The University of Edinburgh's MOCHA database, which provides phonetically-balanced sentences repeated from TIMIT (Zue et al., 1989) uttered by a male and a female speaker (Wrench, 1999), and
- The University of Toronto's TORGO database, from which we select sentences repeated from TIMIT from two females and three males (Rudzicz et al., 2008). (Cerebrally palsied speech, which is the focus of this database, is not included here).

For the following study we use the eight 2D positions common to both databases, namely the upper lip (UL), lower lip (LL), upper incisor (UI), lower incisor (LI), tongue tip (TT), tongue blade (TB), and tongue dorsum (TD). Since these positions are recorded in 3D in TORGO, we project

these onto the midsagittal plane. (Additionally, the MOCHA database provides velum (V) data on this plane, and TORGO provides the left and right lip corners (LL and RL) but these are excluded from study except where noted).

All articulatory data is aligned with its associated acoustic data, which is transformed to Mel-frequency cepstral coefficients (MFCCs). Since the 2D EMA system in MOCHA and the 3D EMA system in TORGO differ in their recording rates, the length of each MFCC frame in each database must differ in order to properly align acoustics with articulation in time. Therefore, each MFCC frame covers 16 ms in the TORGO database, and 32 ms in MOCHA. Phoneme boundaries are determined automatically in the MOCHA database by forced alignment, and by a speech-language pathologist in the TORGO database.

We approximate the tract variable space from the physical space of the articulators, in general, through principal component analysis (PCA) on the latter, and subsequent sigmoid normalization on $[0, 1]$. For example, the LTH tract variable is inferred by calculating the first principal component of the two-dimensional lower incisor (LI) motion in the midsagittal plane, and by normalizing the resulting univariate data through a scaled sigmoid. The VEL variable is inferred similarly from velum (V) EMA data. Tongue tip constriction location and degree (TTCL and TTCD, respectively) are inferred from the 1^{st} and 2^{nd} principal components of tongue tip (TT) EMA data, with TBCL and TBCD inferred similarly from tongue body (TB) data. Finally, the glottis (GLO) is inferred by voicing detection on acoustic energy below 150 Hz (O'Shaughnessy, 2000), lip aperture (LA) is the normalized Euclidean distance between the lips, and lip protrusion (LP) is the normalized 2^{nd} principal component of the midpoint between the lips. All PCA is performed without segmentation of the data. The result is a low-dimensional set of continuous curves describing goal-relevant articulatory variables. Figure 2, for example, shows the degree of the lip aperture (LA) over time for all instances of the */b/* phoneme in the MOCHA database. The relevant articulatory goal of lip closure is evident.

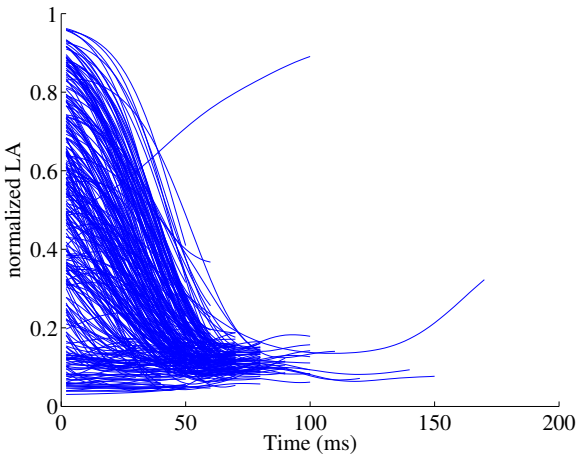


Figure 2: Lip aperture (LA) over time during all MOCHA instances of */b/*.

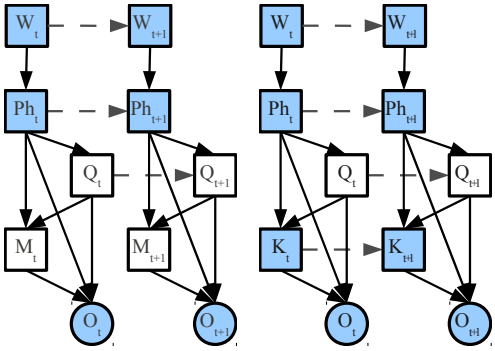
3 Baseline systems

We now turn to the task of speech recognition. Traditional Bayesian learning is restricted to universal or immutable relationships, and is agnostic towards dynamic systems or time-varying relationships. Dynamic Bayes networks (DBNs) are directed acyclic graphs that generalize the powerful stochastic mechanisms of Bayesian representation to temporal sequences. We are free to explicitly provide topological (i.e., dependency) relationships between relevant variables in our models, which can include measurements of tract data.

We examine two baseline systems. The first is the standard acoustic hidden Markov model (HMM) augmented with a bigram language model, as shown in figure 3(a). Here, $W_t \rightarrow W_{t+1}$ represents word transition probabilities, learned by maximum likelihood estimation, and $Ph_t \rightarrow Ph_{t+1}$ represents phoneme transition probabilities whose order is explicitly specified by the relationship $W_t \rightarrow Ph_t$. Likewise, each phoneme Ph conditions the sub-phoneme state, Q_t, whose transition probabilities $Q_t \rightarrow Q_{t+1}$ describe the dynamics within phonemes. The variable M_t refers to hidden Gaussian indices so that the likelihoods of acoustic observations, O_t, are represented by a mixture of 4, 8, 16, or 32 Gaussians for each state and each phoneme. See Murphy (2002) for a further description of this representation.

The second baseline model is the articulatory dynamic Bayes network (DBN-A). This augments the standard acoustic HMM by replacing hidden indices, M_t, with discrete observations of the vocal tract, K_t, as shown in figure 3(b). The pattern of acoustics within each phoneme is dependent on a relatively restricted set of possible articulatory configurations (Roweis, 1999). To find these discrete positions, we obtain k vectors that best de-

scribe the articulatory data according to k-means clustering with the sum-of-squares error function. During training, the DBN variable K_t is set explicitly to the *index* of the mean vector nearest to the current frame of EMA data at time t. In this way, the relationship $K_t \rightarrow O_t$ allows us to learn how discretized articulatory configurations affect acoustics. The training of DBNs involves a specialized version of expectation-maximization, as described in the literature (Murphy, 2002; Ghahramani, 1998). During inference, variables W_t, Ph_t, and K_t become hidden and we marginalize over their possible values when computing their likelihoods. Bigrams are computed by maximum likelihood on lexical annotations in the training data.



(a) HMM (b) DBN-A

Figure 3: Baseline systems: (a) acoustic hidden Markov model and (b) articulatory dynamic Bayes network. Node W_t represents the current word, Ph_t is the current phoneme, Q_t is that phoneme's dynamic state, O_t is the acoustic observation, M_t is the Gaussian mixture component, and K_t is the discretized articulatory configuration. Filled nodes represent observed variables during training, although only O_t is observed during recognition. Square nodes are discrete variables while circular nodes are continuous variables.

4 Switching Kalman filter

Our first experimental system attempts speech recognition given only articulatory data. The true state of the tract variables at time $t-1$ constitutes a 9-dimensional vector, $\mathbf{x}_{t-1}$, of continuous values. Under the task dynamics model of section 2.1, the motions of these tract variables obey critically damped second-order oscillatory relationships. We start with the simplifying assumption of linear dynamics here with allowances for random Gaussian *process noise*, $\mathbf{v}_t$, since articulatory behaviour is non-deterministic. Moreover, we know that EMA recordings are subject to some error (usually less than 1 mm (Yunusova et al., 2009)), so the actual observation at time t, $\mathbf{y}_t$, will not in general be the true position of the articulators. Assuming that the relationship between $\mathbf{y}_t$ and $\mathbf{x}_t$ is also linear, and that the *measurement noise*, $\mathbf{w}_t$, is also Gaussian, then the dynamical articulatory system can be described by

$$\begin{aligned} \mathbf{x}_t &= D_t \mathbf{x}_{t-1} + \mathbf{v}_t \\ \mathbf{y}_t &= C_t \mathbf{x}_t + \mathbf{w}_t. \end{aligned} \tag{2}$$

Eqs. 2 form the basis of the Kalman filter which allows us to use EMA measurements directly, rather than quantized abstractions thereof as in the DBN-A model. Obviously, since articulatory dynamics vary significantly for different goals, we replicate eq. (2) for each phoneme and connect these continuous Kalman filters together with discrete conditioning variables for phoneme and word, resulting in the switching Kalman filter (SKF) model. Here, parameters D_t and $\mathbf{v}_t$ are implicit in the relationship $\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}$, and parameters C_t and $\mathbf{w}_t$ are implicit in $\mathbf{x}_t \rightarrow \mathbf{y}_t$. In this model, observation $\mathbf{y}_t$ is the instantaneous measurements derived from EMA, and $\mathbf{x}_t$ is their true hidden states. These parameters are trained using expectation-maximization, as described in the literature (Murphy, 1998; Deng et al., 2005).

5 Recognition with task dynamics

Our goal is to integrate task dynamics within an ASR system for continuous sentences called TD-ASR. Our approach is to re-rank an N-best list of sentence hypotheses according to a weighted likelihood of their articulatory realizations. For example, if a word sequence $W_i : w_{i,1}\ w_{i,2}\ \ldots\ w_{i,m}$ has likelihoods $L_X(W_i)$ and $L_\Lambda(W_i)$ according to purely acoustic and articulatory interpretations of an utterance, respectively, then its overall score would be

$$L(W_i) = \alpha L_X(W_i) + (1-\alpha) L_\Lambda(W_i) \tag{3}$$

given a weighting parameter α set manually, as in section 6.2. Acoustic likelihoods $L_X(W_i)$ are obtained from Viterbi paths through relevant HMMs in the standard fashion.

5.1 The `TADA` component

In order to obtain articulatory likelihoods, $L_\Lambda(W_i)$, for each word sequence, we first generate articulatory realizations of those sequences according

to task dynamics. To this end, we use components from the open-source TADA system (Nam and Goldstein, 2006), which is a complete implementation of task dynamics. From this toolbox, we use the following components:

- A syllabic dictionary supplemented with the International Speech Lexicon Dictionary (Hasegawa-Johnson and Fleck, 2007). This breaks word sequences W_i into syllable sequences S_i consisting of onsets, nuclei, and coda and covers all of MOCHA and TORGO.
- A syllable-to-gesture lookup table. Given a syllabic sequence, S_i, this table provides the gestural goals necessary to produce those syllables. For example, given the syllable *pub* in figure 1, this table provides the targets for the GLO, VEL, TBCL, and TBCD tract variables, and the parameters for the second-order differential equation, eq. 1, that achieves those goals. These parameters have been empirically tuned by the authors of TADA according to a generic, speaker-independent representation of the vocal tract (Saltzman and Munhall, 1989).
- A component that produces the continuous tract variable paths that produce an utterance. This component takes into account various physiological aspects of human speech production, including intergestural and interarticulator co-ordination and timing (Nam and Saltzman, 2003; Goldstein and Fowler, 2003), and the neutral ("schwa") forces of the vocal tract (Saltzman and Munhall, 1989). This component takes a sequence of gestural goals predicted by the segment-to-gesture lookup table, and produces appropriate paths for each tract variable.

The result of the TADA component is a set of N 9-dimensional articulatory paths, $\mathbf{TV}_i$, necessary to produce the associated word sequences, W_i for $i = 1..N$. Since task dynamics is a prescriptive model and fully deterministic, $\mathbf{TV}_i$ sequences are the *canonical* or default articulatory realizations of the associated sentences. These canonical realizations are independent of our training data, so we transform them in order to more closely resemble the observed articulatory behaviour in our EMA data. Towards this end, we train a switching Kalman filter identical to that in section 4, except the hidden state variable $\mathbf{x}_t$ is replaced by the observed instantaneous *canonical* TVs predicted by TADA. In this way we are explicitly learning a relationship between TADA's task dynamics and human data. Since the lengths of these sequences are generally unequal, we align the articulatory behaviour predicted by TADA with training data from MOCHA and TORGO using standard dynamic time warping (Sakoe and Chiba, 1978). During run-time, the articulatory sequence $\mathbf{y}_t$ most likely to have been produced by the human data given the canonical sequence $\mathbf{TV}_i$ is inferred by the Viterbi algorithm through the SKF model with all other variables hidden. The result is a set of articulatory sequences, $\mathbf{TV}_i^*$, for $i = 1..N$, that represent the predictions of task dynamics that better resemble our data.

5.2 Acoustic-articulatory inversion

In order to estimate the articulatory likelihood of an utterance, we need to evaluate each transformed articulatory sequence, $\mathbf{TV}_i^*$, within probability distributions ranging over all tract variables. These distributions can be inferred using acoustic-articulatory inversion. There are a number of approaches to this task, including vector quantization, and expectation-maximization with Gaussian mixtures (Hogden and Valdez, 2001; Toda et al., 2008). These approaches accurately inferred the *xy* position of articulators to within 0.41 mm and 2.73 mm. Here, we modify the approach taken by Richmond et al. (2003), who estimate probability functions over the 2D midsagittal positions of 7 articulators, given acoustics, with a mixture-density network (MDN). An MDN is essentially a typical discriminative multi-layer neural network whose output consists of the parameters to Gaussian mixtures. Here, each Gaussian mixture describes a probability function over TV positions given the acoustic frame at time t. For example, figure 4 shows an intensity map of the likely values for tongue-tip constriction degree (TTCD) for each frame of acoustics, superimposed with the 'true' trajectory of that TV. Our networks are trained with acoustic and EMA-derived data as described in section 2.2.

5.3 Recognition by reranking

During recognition of a test utterance, a standard acoustic HMM produces word sequence hypotheses, W_i, and associated likelihoods, $L(W_i)$, for $i = 1..N$. The expected canonical motion of the tract variables, $\mathbf{TV}_i$ is then produced by task dynamics

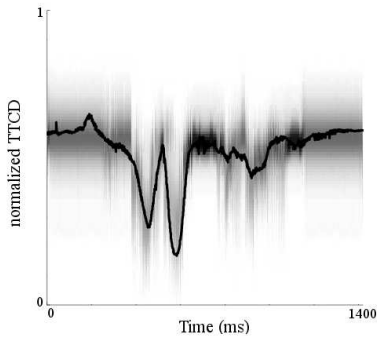


Figure 4: Example probability density of tongue tip constriction degree over time, inferred from acoustics. The true trajectory is superimposed as a black curve.

for each of these word sequences and transformed by an SKF to better match speaker data, giving $\mathbf{TV}_i^*$. The likelihoods of these paths are then evaluated within probability distributions produced by an MDN. The mechanism for producing the articulatory likelihood is shown in figure 5. The overall likelihood, $L(W_i) = \alpha L_X(W_i) + (1-\alpha)L_\Lambda(W_i)$, is then used to produce a final hypothesis list for the given acoustic input.

6 Experiments

Experimental data is obtained from two sources, as described in section 2.2. We procure 1200 sentences from Toronto's TORGO database, and 896 from Edinburgh's MOCHA. In total, there are 460 total unique sentence forms, 1092 total unique word forms, and 11065 total words uttered. Except where noted, all experiments randomly split the data into 90% training and 10% testing sets for 5-cross validation. MOCHA and TORGO data are never combined in a single training set due to differing EMA recording rates. In all cases, models are database-dependent (i.e., all TORGO data is conflated, as is all of MOCHA).

For each of our baseline systems, we calculate the phoneme-error-rate (PER) and word-error-rate (WER) after training. The phoneme-error-rate is calculated according to the proportion of frames of speech incorrectly assigned to the proper phoneme. The word-error-rate is calculated as the sum of insertion, deletion, and substitution errors in the highest-ranked hypothesis divided by the total number of words in the correct orthography. The traditional HMM is compared by varying the number of Gaussians used in the modelling of acoustic observations. Similarly, the DBN-A model is compared by varying the number of discrete quantizations of articulatory configurations, as described in section 3. Results are obtained by direct decoding. The average results across both databases, between which there are no significant differences, are shown in table 1. In all cases the DBN-A model outperforms the HMM, which highlights the benefit of explicitly conditioning acoustic observations on articulatory causes.

System	Parameters	PER (%)	WER (%)
HMM	$\|M\| = 4$	29.3	14.5
	$\|M\| = 8$	27.0	13.9
	$\|M\| = 16$	26.1	10.2
	$\|M\| = 32$	25.6	9.7
DBN-A	$\|K\| = 4$	26.1	13.0
	$\|K\| = 8$	25.2	11.3
	$\|K\| = 16$	24.9	9.8
	$\|K\| = 32$	24.8	9.4

Table 1: Phoneme- and Word-Error-Rate (PER and WER) for different parameterizations of the baseline systems.

	No. of Gaussians			
	1	2	3	4
LTH	−0.28	−0.18	−0.15	−0.11
LA	−0.36	−0.32	−0.30	−0.29
LP	−0.46	−0.44	−0.43	−0.43
GLO	−1.48	−1.30	−1.29	−1.25
TTCD	−1.79	−1.60	−1.51	−1.47
TTCL	−1.81	−1.62	−1.53	−1.49
TBCD	−0.88	−0.79	−0.75	−0.72
TDCL	−0.22	−0.20	−0.18	−0.17

Table 2: Average log likelihood of true tract variable positions in test data, under distributions produced by mixture density networks with varying numbers of Gaussians.

6.1 Efficacy of TD-ASR components

In order to evaluate the whole system, we start by evaluating its parts. First, we test how accurately the mixture-density network (MDN) estimates the position of the articulators given only information from the acoustics available during recognition. Table 2 shows the average log likelihood over each tract variable across both databases. These results are consistent with the state-of-the-art (Toda et al., 2008). In the following experiments, we use MDNs that produce 4 Gaussians.

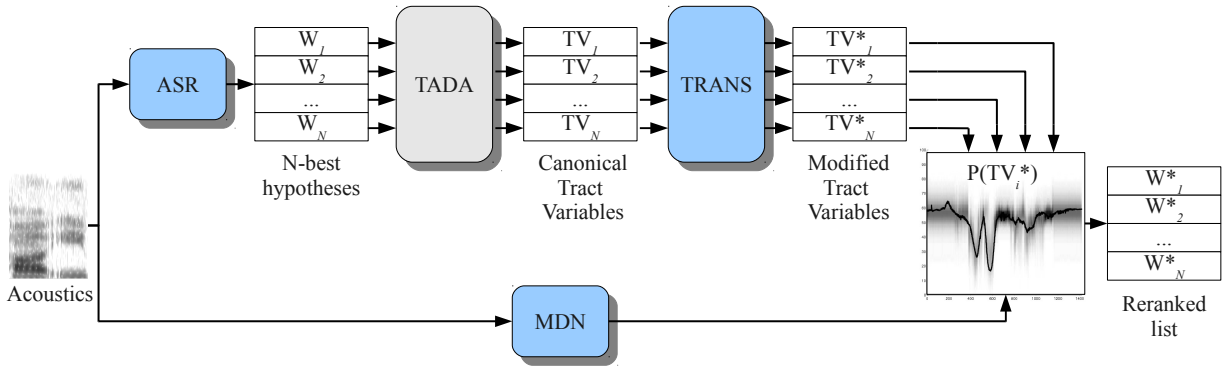


Figure 5: The TD-ASR mechanism for deriving articulatory likelihoods, $L_\Lambda(W_i)$, for each word sequence W_i produced by standard acoustic techniques.

Manner	Canonical	Transformed
approximant	0.19	0.16
fricative	0.37	0.29
nasal*	0.24	0.18
retroflex	0.23	0.19
plosive	0.10	0.08
vowel	0.27	0.25

Table 3: Average difference between predicted tract variables and observed data, on $[0,1]$ scale. (*) Nasals are evaluated only with MOCHA data, since TORGO data lacks velum measurements.

We evaluate how closely transformations to the canonical tract variables predicted by TADA match the data. Namely, we input the known orthography for each test utterance into TADA, obtain the predicted canonical tract variables **TV**, and transform these according to our trained SKF. The resulting predicted and transformed sequences are aligned with our measurements derived from EMA with dynamic time warping. Finally, we measure the average difference between the observed data and the predicted (canonical and transformed) tract variables. Table 3 shows these differences according to the phonological manner of articulation. In all cases the transformed tract variable motion is more accurate, and significantly so at the 95% confidence level for nasal and retroflex phonemes, and at 99% for fricatives. The practical utility of the transformation component is evaluated in its effect on recognition rates, as described below.

6.2 Recognition with TD-ASR

With the performance of the components of TD-ASR better understood, we combine these and study the resulting composite TD-ASR system.

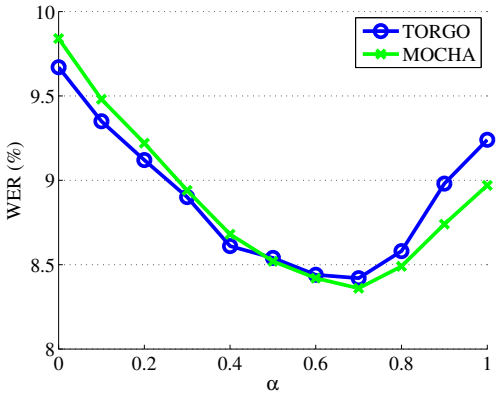


Figure 6: Word-error-rate according to varying α, for both TORGO and MOCHA data.

Figure 6 shows the WER as a function of α with TD-ASR and $N = 4$ hypotheses per utterance. The effect of α is clearly non-monotonic, with articulatory information clearly proving useful. Although systems whose rankings are weighted solely by the articulatory component perform better than the exclusively acoustic systems, the lists available to the former are procured from standard acoustic ASR. Interestingly, the gap between systems trained to the two databases increases as α approaches 1.0. Although this gap is not significant, it may be the result of increased inter-speaker articulatory variation in the TORGO database, which includes more than twice as many speakers as MOCHA.

Figure 7 shows the WER obtained with TD-ASR given varying-length N-best lists and $\alpha = 0.7$. TD-ASR accuracy at $N = 4$ is significantly better than both TD-ASR at $N = 2$ and the baseline approaches of table 1 at the 95% confidence level. However, for $N > 4$ there is a noticeable and systematic worsening of performance.

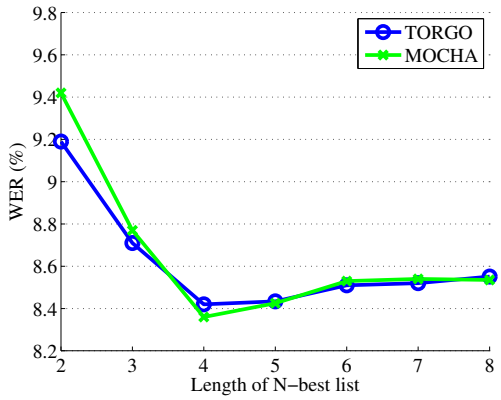


Figure 7: Word-error-rate according to varying lengths of N-best hypotheses used, for both TORGO and MOCHA data.

The optimal parameterization of the TD-ASR model results in an average word-error-rate of 8.43%, which represents a 10.3% relative error reduction over the best parameterization of our baseline models. The SKF model of section 4 differs from the HMM and DBN-A baseline models only in its use of continuous (rather than discrete) hidden dynamics and in its articulatory observations. However, its performance is far more variable, and less conclusive. On the MOCHA database the SKF model had an average of 9.54% WER with a standard deviation of 0.73 over 5 trials, and an average of 9.04% WER with a standard deviation of 0.64 over 5 trials on the TORGO database. Despite the presupposed utility of direct articulatory observations, the SKF system does not perform significantly better than the best DBN-A model.

Finally, the experiments of tables 6 and 7 are repeated with the canonical tract variables passed untransformed to the probability maps generated by the MDNs. Predictably, resulting articulatory likelihoods L_Λ are less representative and increasing their contribution α to the hypothesis reranking does not improve TD-ASR performance significantly, and in some instances worsens it. Although `TADA` is a useful prescriptive model of generic articulation, its use must be tempered with knowledge of inter-speaker variability.

7 Discussion and conclusions

The articulatory medium of speech rarely informs modern speech recognition. We have demonstrated that the use of direct articulatory knowledge can substantially reduce phoneme and word errors in speech recognition, especially if that knowledge is motivated by high-level abstractions of vocal tract behaviour. Task dynamic theory provides a coherent and biologically plausible model of speech production with consequences for phonology (Browman and Goldstein, 1986), neurolinguistics (Guenther and Perkell, 2004), and the evolution of speech and language (Goldstein et al., 2006). We have shown that it is also useful within speech recognition.

We have overcome a conceptual impediment in integrating task dynamics and ASR, which is the former's deterministic nature. This integration is accomplished by stochastically transforming predicted articulatory dynamics and by calculating the likelihoods of these dynamics according to speaker data. However, there are several new avenues for exploration. For example, task dynamics lends itself to more general applications of control theory, including automated self-correction, rhythm, co-ordination, and segmentation (Friedland, 2005). Other high-level questions also remain, such as whether discrete gestures are the correct biological and practical paradigm, whether a purely continuous representation would be more appropriate, and whether this approach generalizes to other languages.

In general, our experiments have revealed very little difference between the use of MOCHA and TORGO EMA data. An *ad hoc* analysis of some of the errors produced by the TD-ASR system found no particular difference between how systems trained to each of these databases recognized nasal phonemes, although only those trained with MOCHA considered velum motion. Other errors common to both sources of data include phoneme insertion errors, normally vowels, which appear to co-occur with some spurious motion of the tongue between segments, especially for longer N-best lists. Despite the relative slow motion of the articulators relative to acoustics, there remains some intermittent noise.

As more articulatory data becomes available and as theories of speech production become more refined, we expect that their combined value to speech recognition will become indispensable.

Acknowledgments

This research is funded by the Natural Sciences and Engineering Research Council and the University of Toronto.

References

Catherine P. Browman and Louis M. Goldstein. 1986. Towards an articulatory phonology. *Phonology Yearbook*, 3:219–252.

Alessandro D'Ausilio, Friedemann Pulvermuller, Paola Salmas, Ilaria Bufalari, Chiara Begliomini, and Luciano Fadiga. 2009. The motor somatotopy of speech perception. *Current Biology*, 19(5):381–385, February.

Jianping Deng, M. Bouchard, and Tet Yeap. 2005. Speech Enhancement Using a Switching Kalman Filter with a Perceptual Post-Filter. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 1, pages 1121–1124, 18-23,.

Bernard Friedland. 2005. *Control System Design: An Introduction to State-Space Methods*. Dover.

Zoubin Ghahramani. 1998. Learning dynamic Bayesian networks. In *Adaptive Processing of Sequences and Data Structures*, pages 168–197. Springer-Verlag.

Louis M. Goldstein and Carol Fowler. 2003. Articulatory phonology: a phonology for public language use. *Phonetics and Phonology in Language Comprehension and Production: Differences and Similarities*.

Louis Goldstein, Dani Byrd, and Elliot Saltzman. 2006. The role of vocal tract gestural action units in understanding the evolution of phonology. In M.A. Arib, editor, *Action to Language via the Mirror Neuron System*, pages 215–249. Cambridge University Press, Cambridge, UK.

Frank H. Guenther and Joseph S. Perkell. 2004. A neural model of speech production and its application to studies of the role of auditory feedback in speech. In Ben Maassen, Raymond Kent, Herman Peters, Pascal Van Lieshout, and Wouter Hulstijn, editors, *Speech Motor Control in Normal and Disordered Speech*, chapter 4, pages 29–49. Oxford University Press, Oxford.

William J. Hardcastle and Nigel Hewlett, editors. 1999. *Coarticulation – Theory, Data, and Techniques*. Cambridge University Press.

Mark Hasegawa-Johnson and Margaret Fleck. 2007. International Speech Lexicon Project.

John Hogden and Patrick Valdez. 2001. A stochastic articulatory-to-acoustic mapping as a basis for speech recognition. In *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference, 2001. IMTC 2001*, volume 2, pages 1105–1110 vol.2.

Katrin Kirchhoff. 1999. *Robust Speech Recognition Using Articulatory Information*. Ph.D. thesis, University of Bielefeld, Germany, July.

Alvin M. Liberman and Ignatius G. Mattingly. 1985. The motor theory of speech perception revised. *Cognition*, 21:1–36.

Konstantin Markov, Jianwu Dang, and Satoshi Nakamura. 2006. Integration of articulatory and spectrum features based on the hybrid HMM/BN modeling framework. *Speech Communication*, 48(2):161–175, February.

Kevin Patrick Murphy. 1998. Switching Kalman Filters. Technical report.

Kevin Patrick Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California at Berkeley.

Hosung Nam and Louis Goldstein. 2006. TADA (TAsk Dynamics Application) manual.

Hosung Nam and Elliot Saltzman. 2003. A competitive, coupled oscillator model of syllable structure. In *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS 2003)*, pages 2253–2256, Barcelona, Spain.

Douglas O'Shaughnessy. 2000. *Speech Communications – Human and Machine*. IEEE Press, New York, NY, USA.

Korin Richmond, Simon King, and Paul Taylor. 2003. Modelling the uncertainty in recovering articulation from acoustics. *Computer Speech and Language*, 17:153–172.

Sam T. Roweis. 1999. *Data Driven Production Models for Speech Processing*. Ph.D. thesis, California Institute of Technology, Pasadena, California.

Frank Rudzicz, Pascal van Lieshout, Graeme Hirst, Gerald Penn, Fraser Shein, and Talya Wolff. 2008. Towards a comparative database of dysarthric articulation. In *Proceedings of the eighth International Seminar on Speech Production (ISSP'08)*, Strasbourg France, December.

Frank Rudzicz. 2009. Applying discretized articulatory knowledge to dysarthric speech. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP09)*, Taipei, Taiwan, April.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26, February.

Elliot L. Saltzman and Kevin G. Munhall. 1989. A dynamical approach to gestural patterning in speech production. *Ecological Psychology*, 1(4):333–382.

Elliot M. Saltzman, 1986. *Task dynamic co-ordination of the speech articulators: a preliminary model*, pages 129–144. Springer-Verlag.

Tomoki Toda, Alan W. Black, and Keiichi Tokuda. 2008. Statistical mapping between articulatory movements and acoustic spectrum using a Gaussian mixture model. *Speech Communication*, 50(3):215–227, March.

Alan Wrench. 1999. The MOCHA-TIMIT articulatory database, November.

Yana Yunusova, Jordan R. Green, and Antje Mefferd. 2009. Accuracy Assessment for AG500, Electromagnetic Articulograph. *Journal of Speech, Language, and Hearing Research*, 52:547–555, April.

Victor Zue, Stephanie Seneff, and James Glass. 1989. Speech Database Development: TIMIT and Beyond. In *Proceedings of ESCA Tutorial and Research Workshop on Speech Input/Output Assessment and Speech Databases (SIOA-1989)*, volume 2, pages 35–40, Noordwijkerhout, The Netherlands.

Learning to Adapt to Unknown Users: Referring Expression Generation in Spoken Dialogue Systems

Srinivasan Janarthanam
School of Informatics
University of Edinburgh
s.janarthanam@ed.ac.uk

Oliver Lemon
Interaction Lab
Mathematics and Computer Science (MACS)
Heriot-Watt University
o.lemon@hw.ac.uk

Abstract

We present a data-driven approach to learn user-adaptive referring expression generation (REG) policies for spoken dialogue systems. Referring expressions can be difficult to understand in technical domains where users may not know the technical 'jargon' names of the domain entities. In such cases, dialogue systems must be able to model the user's (lexical) domain knowledge and use appropriate referring expressions. We present a reinforcement learning (RL) framework in which the system learns REG policies which can adapt to unknown users online. Furthermore, unlike supervised learning methods which require a large corpus of expert adaptive behaviour to train on, we show that effective adaptive policies can be learned from a small dialogue corpus of non-adaptive human-machine interaction, by using a RL framework and a statistical user simulation. We show that in comparison to adaptive hand-coded baseline policies, the learned policy performs significantly better, with an 18.6% average increase in adaptation accuracy. The best learned policy also takes less dialogue time (average 1.07 min less) than the best hand-coded policy. This is because the learned policies can adapt online to changing evidence about the user's domain expertise.

1 Introduction

We present a reinforcement learning (Sutton and Barto, 1998) framework to learn user-adaptive referring expression generation policies from data-driven user simulations. A user-adaptive REG policy allows the system to choose appropriate expressions to refer to domain entities in a dialogue setting. For instance, in a technical support conversation, the system could choose to use more technical terms with an expert user, or to use more descriptive and general expressions with novice users, and a mix of the two with intermediate users of various sorts (see examples in Table 1).

Jargon: Please plug one end of the `broadband cable` into the `broadband filter`.
Descriptive: Please plug one end of the `thin white cable with grey ends` into the `small white box`.

Table 1: Referring expression examples for 2 entities (from the corpus)

In natural human-human conversations, dialogue partners learn about each other and adapt their language to suit their domain expertise (Issacs and Clark, 1987). This kind of adaptation is called `Alignment through Audience Design` (Clark and Murphy, 1982; Bell, 1984). We assume that users are mostly unknown to the system and therefore that a spoken dialogue system (SDS) must be capable of observing the user's dialogue behaviour, modelling his/her domain knowledge, and adapting accordingly, just like human interlocutors. Rule-based and supervised learning approaches to user adaptation in SDS have been proposed earlier (Cawsey, 1993; Akiba and Tanaka, 1994). However, such methods require expensive resources such as domain experts to hand-code the rules, or a corpus of expert-layperson interactions to train on. In contrast, we present a corpus-driven framework using which a user-adaptive REG policy can be learned using RL from a small corpus of non-adaptive human-machine interaction.

We show that these learned policies perform better than simple hand-coded adaptive policies in terms of accuracy of adaptation and dialogue

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 69–78,
Uppsala, Sweden, 11-16 July 2010.

time. We also compared the performance of policies learned using a hand-coded rule-based simulation and a data-driven statistical simulation and show that data-driven simulations produce better policies than rule-based ones.

In section 2, we present some of the related work. Section 3 presents the dialogue data that we used to train the user simulation. Section 4 and section 5 describe the dialogue system framework and the user simulation models. In section 6, we present the training and in section 7, we present the evaluation for different REG policies.

2 Related work

There are several ways in which natural language generation (NLG) systems adapt to users. Some of them adapt to a user's goals, preferences, environment and so on. Our focus in this study is restricted to the user's lexical domain expertise. Several NLG systems adapt to the user's domain expertise at different levels of generation - text planning (Paris, 1987), complexity of instructions (Dale, 1989), referring expressions (Reiter, 1991), and so on. Some dialogue systems, such as COMET, have also incorporated NLG modules that present appropriate levels of instruction to the user (McKeown et al., 1993). However, in all the above systems, the user's knowledge is assumed to be accurately represented in an initial user model using which the system adapts its language. In contrast to all these systems, our adaptive REG policy knows nothing about the user when the conversation starts.

Rule-based and supervised learning approaches have been proposed to learn and adapt during the conversation dynamically. Such systems learned from the user at the start and later adapted to the domain knowledge of the users. However, they either require expensive expert knowledge resources to hand-code the inference rules (Cawsey, 1993) or large corpus of expert-layperson interaction from which adaptive strategies can be learned and modelled, using methods such as Bayesian networks (Akiba and Tanaka, 1994). In contrast, we present an approach that learns in the absence of these expensive resources. It is also not clear how supervised and rule-based approaches choose between when to seek more information and when to adapt. In this study, we show that using reinforcement learning this decision is learned automatically.

Reinforcement Learning (RL) has been successfully used for learning dialogue management policies since (Levin et al., 1997). The learned policies allow the dialogue manager to optimally choose appropriate dialogue acts such as instructions, confirmation requests, and so on, under uncertain noise or other environment conditions. There have been recent efforts to learn information presentation and recommendation strategies using reinforcement learning (Rieser and Lemon, 2009; Hernandez et al., 2003; Rieser and Lemon, 2010), and joint optimisation of Dialogue Management and NLG using hierarchical RL has been proposed by (Lemon, 2010). In contrast, we present a framework to learn to choose appropriate referring expressions based on a user's domain knowledge. Earlier, we reported a proof-of-concept work using a hand-coded rule-based user simulation (Janarthanam and Lemon, 2009c).

3 The Wizard-of-Oz Corpus

We use a corpus of technical support dialogues collected from real human users using a Wizard-of-Oz method (Janarthanam and Lemon, 2009b). The corpus consists of 17 dialogues from users who were instructed to physically set up a home broadband connection using objects like a wireless modem, cables, filters, etc. They listened to the instructions from the system and carried them out using the domain objects laid in front of them. The human 'wizard' played the role of only an interpreter who would understand what the user said and annotate it as a dialogue act. The set-up examined the effect of using three types of referring expressions (jargon, descriptive, and tutorial), on the users.

Out of the 17 dialogues, 6 used a jargon strategy, 6 used a descriptive strategy, and 5 used a tutorial strategy[1]. The task had reference to 13 domain entities, mentioned repeatedly in the dialogue. In total, there are 203 jargon, 202 descriptive and 167 tutorial referring expressions. Interestingly, users who weren't acquainted with the domain objects requested clarification on some of the referring expressions used. The dialogue exchanges between the user and system were logged in the form of dialogue acts and the system's choices of referring expressions. Each user's knowledge of domain entities was recorded both before and after the task and each user's interac-

[1]The tutorial strategy uses both jargon and descriptive expressions together.

tions with the environment were recorded. We use the dialogue data, pre-task knowledge tests, and the environment interaction data to train a user simulation model. Pre and post-task test scores were used to model the learning behaviour of the users during the task (see section 5).

The corpus also recorded the time taken to complete each dialogue task. We used these data to build a regression model to calculate total dialogue time for dialogue simulations. The strategies were never mixed (with some jargon, some descriptive and some tutorial expressions) within a single conversation. Therefore, please note that the strategies used for data collection were *not adaptive* and the human 'wizard' has no role in choosing which referring expression to present to the user. Due to this fact, no user score regarding adaptation was collected. We therefore measure adaptation objectively as explained in section 6.1.

4 The Dialogue System

In this section, we describe the different modules of the dialogue system. The interaction between the different modules is shown in figure 1 (in learning mode). The dialogue system presents the user with instructions to setup a broadband connection at home. In the Wizard of Oz setup, the system and the user interact using speech. However, in our machine learning setup, they interact at the abstract level of dialogue actions and referring expressions. Our objective is to learn to choose the appropriate referring expressions to refer to the domain entities in the instructions.

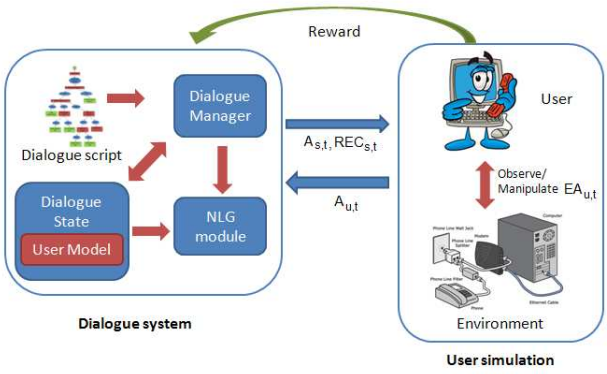


Figure 1: System User Interaction (learning)

4.1 Dialogue Manager

The dialogue manager identifies the next instruction (dialogue act) to give to the user based on the dialogue management policy π_{dm}. Since, in this study, we focus only on learning the REG policy, the dialogue management is coded in the form of a finite state machine. In this dialogue task, the system provides two kinds of instructions - observation and manipulation. For observation instructions, users observe the environment and report back to the system, and for the manipulation instructions (such as plugging in a cable in to a socket), they manipulate the domain entities in the environment. When the user carries out an instruction, the system state is updated and the next instruction is given. Sometimes, users do not understand the referring expressions used by the system and then ask for clarification. In such cases, the system provides clarification on the referring expression ($provide\_clar$), which is information to enable the user to associate the expression with the intended referent. The system action $A_{s,t}$ (t denoting turn, s denoting system) is therefore to either give the user the next instruction or a clarification. When the user responds in any other way, the instruction is simply repeated. The dialogue manager is also responsible for updating and managing the system state $S_{s,t}$ (see section 4.2). The system interacts with the user by passing both the system action $A_{s,t}$ and the referring expressions $REC_{s,t}$ (see section 4.3).

4.2 The dialogue state

The dialogue state $S_{s,t}$ is a set of variables that represent the current state of the conversation. In our study, in addition to maintaining an overall dialogue state, the system maintains a user model $UM_{s,t}$ which records the initial domain knowledge of the user. It is a dynamic model that starts with a state where the system does not have any idea about the user. As the conversation progresses, the dialogue manager records the evidence presented to it by the user in terms of his dialogue behaviour, such as asking for clarification and interpreting jargon. Since the model is updated according to the user's behaviour, it may be inaccurate if the user's behaviour is itself uncertain. So, when the user's behaviour changes (for instance, from novice to expert), this is reflected in the user model during the conversation. Hence, unlike previous studies mentioned in section 2, the user model used in this system is not always an accurate model of the user's knowledge and reflects a level of uncertainty about the user.

Each jargon referring expression x is represented by a three valued variable in the dialogue state: user_knows_x. The three values that each variable takes are yes, no, not_sure. The variables are updated using a simple user model update algorithm. Initially each variable is set to not_sure. If the user responds to an instruction containing the referring expression x with a clarification request, then user_knows_x is set to no. Similarly, if the user responds with appropriate information to the system's instruction, the dialogue manager sets user_knows_x is set to yes.

The dialogue manager updates the variables concerning the referring expressions used in the current system utterance appropriately after the user's response each turn. The user may have the capacity to learn jargon. However, only the user's initial knowledge is recorded. This is based on the assumption that an estimate of the user's knowledge helps to predict the user's knowledge of the rest of the referring expressions. Another issue concerning the state space is its size. Since, there are 13 entities and we only model the jargon expressions, the state space size is 3^{13}.

4.3 REG module

The REG module is a part of the NLG module whose task is to identify the list of domain entities to be referred to and to choose the appropriate referring expression for each of the domain entities for each given dialogue act. In this study, we focus only on the production of appropriate referring expressions to refer to domain entities mentioned in the dialogue act. It chooses between the two types of referring expressions - jargon and descriptive. For example, the domain entity *broadband_filter* can be referred to using the jargon expression "broadband filter" or using the descriptive expression "small white box"[2]. We call this the act of choosing the *REG action*. The tutorial strategy was not investigated here since the corpus analysis showed tutorial utterances to be very time consuming. In addition, they do not contribute to the adaptive behaviour of the system.

The REG module operates in two modes - learning and evaluation. In the learning mode, the REG module is the learning agent. The REG module learns to associate dialogue states with optimal REG actions. This is represented by a REG policy $\pi_{reg} : UM_{s,t} \rightarrow REC_{s,t}$, which maps the states of the dialogue (user model) to optimal REG actions. The referring expression choices $REC_{s,t}$ is a set of pairs identifying the referent R and the type of expression T used in the current system utterance. For instance, the pair $(broadband\_filter, desc)$ represents the descriptive expression "small white box".

$$REC_{s,t} = \{(R_1, T_1), ..., (R_n, T_n)\}$$

In the evaluation mode, a trained REG policy interacts with unknown users. It consults the learned policy π_{reg} to choose the referring expressions based on the current user model.

5 User Simulations

In this section, we present user simulation models that simulate the dialogue behaviour of a real human user. These external simulation models are different from internal user models used by the dialogue system. In particular, our model is the first to be sensitive to a system's choices of referring expressions. The simulation has a statistical distribution of in-built knowledge profiles that determines the dialogue behaviour of the user being simulated. If the user does not know a referring expression, then he is more *likely* to request clarification. If the user is able to interpret the referring expressions and identify the references then he is more likely to follow the system's instruction. This behaviour is simulated by the action selection models described below.

Several user simulation models have been proposed for use in reinforcement learning of dialogue policies (Georgila et al., 2005; Schatzmann et al., 2006; Schatzmann et al., 2007; Ai and Litman, 2007). However, they are suited only for learning dialogue management policies, and not natural language generation policies. Earlier, we presented a two-tier simulation trained on data precisely for REG policy learning (Janarthanam and Lemon, 2009a). However, it is not suited for training on small corpus like the one we have at our disposal. In contrast to the earlier model, we now condition the clarification requests on the referent class rather than the referent itself to handle data sparsity problem.

The user simulation (US) receives the system action $A_{s,t}$ and its referring expression choices $REC_{s,t}$ at each turn. The US responds with a user action $A_{u,t}$ (u denoting user). This can either be a clarification request (cr) or an instruction

[2] We will use italicised forms to represent the domain entities (e.g. *broadband_filter*) and double quotes to represent the referring expressions (e.g. "broadband filter").

response (ir). We used two kinds of action selection models: corpus-driven statistical model and hand-coded rule-based model.

5.1 Corpus-driven action selection model

In the corpus-driven model, the US produces a clarification request cr based on the class of the referent $C(R_i)$, type of the referring expression T_i, and the current domain knowledge of the user for the referring expression $DK_{u,t}(R_i, T_i)$. Domain entities whose jargon expressions raised clarification requests in the corpus were listed and those that had more than the mean number of clarification requests were classified as `difficult` and others as `easy` entities (for example, "power adaptor" is `easy` - all users understood this expression, "broadband filter" is `difficult`). Clarification requests are produced using the following model.

$$P(A_{u,t} = cr(R_i, T_i)|C(R_i), T_i, DK_{u,t}(R_i, T_i))$$
$$\text{where } (R_i, T_i) \in REC_{s,t}$$

One should note that the actual literal expression is not used in the transaction. Only the entity that it is referring to (R_i) and its type (T_i) are used. However, the above model simulates the process of interpreting and resolving the expression and identifying the domain entity of interest in the instruction. The user identification of the entity is signified when there is no clarification request produced (i.e. $A_{u,t} = none$). When no clarification request is produced, the environment action $EA_{u,t}$ is generated using the following model.

$$P(EA_{u,t}|A_{s,t}) \text{ if } A_{u,t} != cr(R_i, T_i)$$

Finally, the user action is an instruction response which is determined by the system action $A_{s,t}$. Instruction responses can be different in different conditions. For an observe and report instruction, the user issues a $provide\_info$ action and for a manipulation instruction, the user responds with an $acknowledgement$ action and so on.

$$P(A_{u,t} = ir|EA_{u,t}, A_{s,t})$$

All the above models were trained on our corpus data using *maximum likelihood estimation* and smoothed using a variant of *Witten-Bell discounting*. According to the data, clarification requests are much more likely when jargon expressions are used to refer to the referents that belong to the `difficult` class and which the user doesn't know about. When the system uses expressions that the user knows, the user generally responds to the instruction given by the system. These user simulation models have been evaluated and found to produce behaviour that is very similar to the original corpus data, using the Kullback-Leibler divergence metric (Cuayahuitl, 2009).

livebox $= 1$	*power_adaptor* $= 1$
wall_phone_socket $= 1$	*broadband_filter* $= 0$
broadband_cable $= 0$	*ethernet_cable* $= 1$
lb_power_light $= 1$	*lb_power_socket* $= 1$
lb_broadband_light $= 0$	*lb_ethernet_light* $= 0$
lb_adsl_socket $= 0$	*lb_ethernet_socket* $= 0$
pc_ethernet_socket $= 1$	

Table 2: Domain knowledge: an Intermediate User

5.2 Rule-based action selection model

We also built a rule-based simulation using the above models but where some of the parameters were set manually instead of estimated from the data. The purpose of this simulation is to investigate how learning with a data-driven statistical simulation compares to learning with a simple hand-coded rule-based simulation. In this simulation, the user *always* asks for a clarification when he does not know a jargon expression (regardless of the class of the referent) and never does this when he knows it. This enforces a stricter, more consistent behaviour for the different knowledge patterns, which we hypothesise should be easier to learn to adapt to, but may lead to less robust REG policies.

5.3 User Domain knowledge

The user domain knowledge is initially set to one of several models at the start of every conversation. The models range from novices to experts which were identified from the corpus using `k-means` clustering. The initial knowledge base ($DK_{u,initial}$) for an intermediate user is shown in table 2. A novice user knows only "power adaptor", and an expert knows all the jargon expressions. We assume that users can interpret the descriptive expressions and resolve their references. Therefore, they are not explicitly represented. We only code the user's knowledge of jargon expressions. This is represented by a boolean variable for each domain entity.

Corpus data shows that users can learn jargon expressions during the conversation. The user's domain knowledge DK_u is modelled to be dynamic and is updated during the conversation. Based on our data, we found that when presented with clarification on a jargon expression, users always learned the jargon.

$$\text{if } A_{s,t} = provide\_clar(R_i, T_i)$$
$$DK_{u,t+1}(R_i, T_i) \leftarrow 1$$

Users also learn when jargon expressions are repeatedly presented to them. Learning by repetition follows the pattern of a learning curve - the greater the number of repetitions $\#(R_i, T_i)$, the higher the likelihood of learning. This is modelled stochastically based on repetition using the parameter $\#(R_i, T_i)$ as follows (where $(R_i, T_i) \in REC_{s,t}$) .

$$P(DK_{u,t+1}(R_i, T_i) \leftarrow 1 | \#(R_i, T_i))$$

The final state of the user's domain knowledge ($DK_{u,final}$) may therefore be different from the initial state ($DK_{u,initial}$) due to the learning effect produced by the system's use of jargon expressions. In most studies done previously, the user's domain knowledge is considered to be static. However in real conversation, we found that the users nearly always learned jargon expressions from the system's utterances and clarifications.

6 Training

The REG module was trained (operated in learning mode) using the above simulations to learn REG policies that select referring expressions based on the user expertise in the domain. As shown in figure 1, the learning agent (REG module) is given a reward at the end of every dialogue. During the training session, the learning agent explores different ways to maximize the reward. In this section, we discuss how to code the learning agent's goals as reward. We then discuss how the reward function is used to train the learning agent.

6.1 Reward function

A reward function generates a numeric reward for the learning agent's actions. It gives high rewards to the agent when the actions are favourable and low rewards when they are not. In short, the reward function is a representation of the goal of the agent. It translates the agent's actions into a scalar value that can be maximized by choosing the right action sequences.

We designed a reward function for the goal of adapting to each user's domain knowledge. We present the Adaptation Accuracy score AA that calculates how accurately the agent chose the expressions for each referent r, with respect to the user's knowledge. Appropriateness of an expression is based on the user's knowledge of the expression. So, when the user knows the jargon expression for r, the appropriate expression to use is jargon, and if s/he doesn't know the jargon, an descriptive expression is appropriate. Although the user's domain knowledge is dynamically changing due to learning, we base appropriateness on the initial state, because our objective is to adapt to the initial state of the user $DK_{u,initial}$. However, in reality, designers might want their system to account for user's changing knowledge as well. We calculate accuracy per referent RA_r as the ratio of number of appropriate expressions to the total number of instances of the referent in the dialogue. We then calculate the overall mean accuracy over all referents as shown below.

$$RA_r = \frac{\#(appropriate\_expressions(r))}{\#(instances(r))}$$
$$Adaptation Accuracy AA = \frac{1}{\#(r)} \Sigma_r RA_r$$

Note that this reward is computed at the end of the dialogue (it is a 'final' reward), and is then back-propagated along the action sequence that led to that final state. Thus the reward can be computed for each system REG action, without the system having access to the user's initial domain knowledge while it is learning a policy.

Since the agent starts the conversation with no knowledge about the user, it may try to use more exploratory moves to learn about the user, although they may be inappropriate. However, by measuring accuracy to the initial user state, the agent is encouraged to restrict its exploratory moves and start predicting the user's domain knowledge as soon as possible. The system should therefore ideally explore less and adapt more to increase accuracy. The above reward function returns 1 when the agent is completely accurate in adapting to the user's domain knowledge and it returns 0 if the agent's REC choices were completely inappropriate. Usually during learning, the reward value lies between these two extremes and the agent tries to maximize it to 1.

6.2 Learning

The REG module was trained in learning mode using the above reward function using the SHARSHA reinforcement learning algorithm (with linear function approximation) (Shapiro and Langley, 2002). This is a hierarchical variant of SARSA, which is an on-policy learning algorithm that updates the current behaviour policy (see (Sutton and Barto, 1998)). The training produced approx. 5000 dialogues. Two types of simulations were used as described above: Data-driven and Hand-coded. Both user simulations were calibrated to produce three types of users: Novice, Int2 (intermediate) and Expert, randomly but with equal probability. Novice users knew just one jargon expression, Int2 knew seven, and Expert users knew all thirteen jargon expressions. There was an underlying pattern in these knowledge profiles. For example, Intermediate users were those who knew the commonplace domain entities but not those specific to broadband connection. For instance, they knew "ethernet cable" and "pc ethernet socket" but not "broadband filter" and "broadband cable".

Initially, the REG policy chooses randomly between the referring expression types for each domain entity in the system utterance, irrespective of the user model state. Once the referring expressions are chosen, the system presents the user simulation with both the dialogue act and referring expression choices. The choice of referring expression affects the user's dialogue behaviour which in turn makes the dialogue manager update the user model. For instance, choosing a jargon expression could evoke a clarification request from the user, which in turn prompts the dialogue manager to update the user model with the new information that the user is ignorant of the particular expression. It should be noted that using a jargon expression is an *information seeking* move which enables the REG module to estimate the user's knowledge level. The same process is repeated for every dialogue instruction. At the end of the dialogue, the system is rewarded based on its choices of referring expressions. If the system chooses jargon expressions for novice users or descriptive expressions for expert users, penalties are incurred and if the system chooses REs appropriately, the reward is high. On the one hand, those actions that fetch more reward are reinforced, and on the other hand, the agent tries out new state-action combinations to explore the possibility of greater rewards. Over time, it stops exploring new state-action combinations and exploits those actions that contribute to higher reward. The REG module learns to choose the appropriate referring expressions based on the user model in order to maximize the overall adaptation accuracy.

Figure 2 shows how the agent learns using the data-driven (**Learned DS**) and hand-coded simulations (**Learned HS**) during training. It can be seen in the figure 2 that towards the end the curve plateaus signifying that learning has converged.

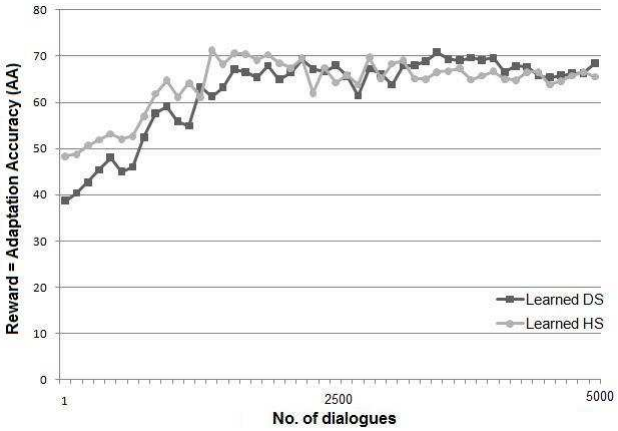


Figure 2: Learning curves - Training

7 Evaluation

In this section, we present the evaluation metrics used, the baseline policies that were hand-coded for comparison, and the results of evaluation.

7.1 Metrics

In addition to the adaptation accuracy mentioned in section 6.1, we also measure other parameters from the conversation in order to show how learned adaptive policies compare with other policies on other dimensions. We calculate the time taken ($Time$) for the user to complete the dialogue task. This is calculated using a regression model from the corpus based on number of words, turns, and mean user response time. We also measure the (normalised) learning gain (LG) produced by using unknown jargon expressions. This is calculated using the pre and post scores from the user domain knowledge (DK_u) as follows.

$$Learning\ Gain\ LG = \frac{Post-Pre}{1-Pre}$$

7.2 Baseline REG policies

In order to compare the performance of the learned policy with hand-coded REG policies, three simple rule-based policies were built. These were built in the absence of expert domain knowledge and a expert-layperson corpus.

- Jargon: Uses jargon for all referents by default. Provides clarifications when requested.
- Descriptive: Uses descriptive expressions for all referents by default.
- Switching: This policy starts with jargon expressions and continues using them until the user requests for clarification. It then switches to descriptive expressions and continues to use them until the user complains. In short, it switches between the two strategies based on the user's responses.

All the policies exploit the user model in subsequent references after the user's knowledge of the expression has been set to either `yes` or `no`. Therefore, although these policies are simple, they do adapt to a certain extent, and are reasonable baselines for comparison in the absence of expert knowledge for building more sophisticated baselines.

7.3 Results

The policies were run under a testing condition (where there is no policy learning or exploration) using a data-driven simulation calibrated to simulate 5 different user types. In addition to the three users - Novice, Expert and Int2, from the training simulations, two other intermediate users (Int1 and Int3) were added to examine how well each policy handles unseen user types. The REG module was operated in evaluation mode to produce around 200 dialogues per policy distributed over the 5 user groups.

Overall performance of the different policies in terms of $Adaptation\ Accuracy\ (AA)$, $Time$ and $Learning\ Gain\ (LG)$ are given in Table 3. Figure 3 shows how each policy performs in terms of accuracy on the 5 types of users.

We found that the Learned DS policy (i.e. learned with the data-driven user simulation) is the most accurate (Mean = 79.70, SD = 10.46) in terms of adaptation to each user's initial state of domain knowledge. Also, it is the only policy that has more or less the same accuracy scores

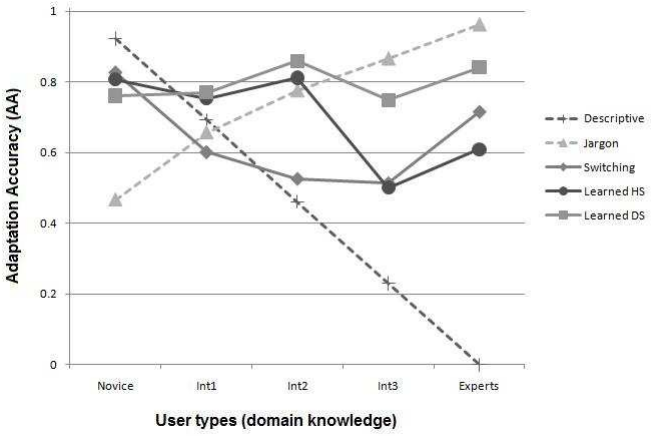


Figure 3: Evaluation - Adaptation Accuracy

Policies	AA	Time T	LG
Descriptive	46.15	7.44	0
Jargon	74.54	9.15*	0.97*
Switching	62.47	7.48	0.30
Learned HS	69.67	7.52	0.33
Learned DS	**79.70***	8.08*	0.63*

* Significantly different from all others ($p < 0.05$).

Table 3: Evaluation on 5 user types

over all different user types (see figure 3). It should also be noted that the it generalised well over user types (Int1 and Int3) which were unseen in training. Learned DS policy outperforms all other policies: Learned HS (Mean = 69.67, SD = 14.18), Switching (Mean = 62.47, SD = 14.18), Jargon (Mean = 74.54, SD = 17.9) and Descriptive (Mean = 46.15, SD = 33.29). The differences between the accuracy (AA) of the Learned DS policy and all other policies were statistically significant with $p < 0.05$ (using a two-tailed paired t-test). Although Learned HS policy is similar to the Learned DS policy, as shown in the learning curves in figure 2, it does not perform as well when confronted with users types that it did not encounter during training. The Switching policy, on the other hand, quickly switches its strategy (sometimes erroneously) based on the user's clarification requests but does not adapt appropriately to evidence presented later during the conversation. Sometimes, this policy switches erroneously because of the uncertain user behaviours. In contrast, learned policies continuously adapt to new evidence. The Jargon policy performs better than

the Learned HS and Switching policies. This because the system can learn more about the user by using more jargon expressions and then use that knowledge for adaptation for known referents. However, it is not possible for this policy to predict the user's knowledge of unseen referents. The Learned DS policy performs better than the Jargon policy, because it is able to accurately predict the user's knowledge of referents unseen in the dialogue so far.

The learned policies are a little more time-consuming than the Switching and Descriptive policies but compared to the Jargon policy, Learned DS takes 1.07 minutes less time. This is because learned policies use a few jargon expressions (giving rise to clarification requests) to learn about the user. On the other hand, the Jargon policy produces more user learning gain because of the use of more jargon expressions. Learned policies compensate on time and learning gain in order to predict and adapt well to the users' knowledge patterns. This is because the training was optimized for accuracy of adaptation and not for learning gain or time taken. The results show that using our RL framework, REG policies can be learned using data-driven simulations, and that such a policy can predict and adapt to a user's knowledge pattern more accurately than policies trained using hand-coded rule-based simulations and hand-coded baseline policies.

7.4 Discussion

The learned policies explore the user's expertise and predict their knowledge patterns, in order to better choose expressions for referents unseen in the dialogue so far. The system learns to identify the patterns of knowledge in the users with a little exploration (information seeking moves). So, when it is provided with a piece of evidence (e.g. the user knows "broadband filter"), it is able to accurately estimate unknown facts (e.g. the user might know "broadband cable"). Sometimes, its choices are wrong due to incorrect estimation of the user's expertise (due to stochastic behaviour of the users). In such cases, the incorrect adaptation move can be considered to be an information seeking move. This helps further adaptation using the new evidence. By continuously using this "seek-predict-adapt" approach, the system adapts dynamically to different users. Therefore, with a little information seeking and better prediction, the learned policies are able to better adapt to users with different domain expertise.

In addition to adaptation, learned policies learn to identify when to seek information from the user to populate the user model (which is initially set to `not_sure`). It should be noted that the system cannot adapt unless it has some information about the user and therefore needs to decisively seek information by using jargon expressions. If it seeks information all the time, it is not adapting to the user. The learned policies therefore learn to trade-off between information seeking moves and adaptive moves in order to maximize the overall adaptation accuracy score.

8 Conclusion

In this study, we have shown that user-adaptive REG policies can be learned from a small corpus of non-adaptive dialogues between a dialogue system and users with different domain knowledge levels. We have shown that such adaptive REG policies learned using a RL framework adapt to unknown users better than simple hand-coded policies built without much input from domain experts or from a corpus of expert-layperson adaptive dialogues. The learned, adaptive REG policies learn to trade off between adaptive moves and information seeking moves automatically to maximize the overall adaptation accuracy. Learned policies start the conversation with information seeking moves, learn a little about the user, and start adapting dynamically as the conversation progresses. We have also shown that a data-driven statistical user simulation produces better policies than a simple hand-coded rule-based simulation, and that the learned policies generalise well to unseen users.

In future work, we will evaluate the learned policies with real users to examine how well they adapt, and examine how real users evaluate them (subjectively) in comparison to baselines. Whether the learned policies perform better or as well as a hand-coded policy painstakingly crafted by a domain expert (or learned using supervised methods from an expert-layperson corpus) is an interesting question that needs further exploration. Also, it would also be interesting to make the learned policy account for the user's learning behaviour and adapt accordingly.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (CLASSiC project www.classic-project.org) and from the EPSRC, project no. EP/G069840/1.

References

H. Ai and D. Litman. 2007. Knowledge consistent user simulations for dialog systems. In *Proceedings of Interspeech 2007, Antwerp, Belgium*.

T. Akiba and H. Tanaka. 1994. A Bayesian approach for User Modelling in Dialogue Systems. In *Proceedings of the 15th conference on Computational Linguistics - Volume 2, Kyoto*.

A. Bell. 1984. Language style as audience design. *Language in Society*, 13(2):145–204.

A. Cawsey. 1993. User Modelling in Interactive Explanations. *User Modeling and User-Adapted Interaction*, 3(3):221–247.

H. H. Clark and G. L. Murphy. 1982. Audience design in meaning and reference. In J. F. LeNy and W. Kintsch, editors, *Language and comprehension*. Amsterdam: North-Holland.

H. Cuayahuitl. 2009. *Hierarchical Reinforcement Learning for Spoken Dialogue Systems*. Ph.D. thesis, University of Edinburgh, UK.

R. Dale. 1989. Cooking up referring expressions. In *Proc. ACL-1989*.

K. Georgila, J. Henderson, and O. Lemon. 2005. Learning User Simulations for Information State Update Dialogue Systems. In *Proc of Eurospeech/Interspeech*.

F. Hernandez, E. Gaudioso, and J. G. Boticario. 2003. A Multiagent Approach to Obtain Open and Flexible User Models in Adaptive Learning Communities. In *User Modeling 2003*, volume 2702/2003 of *LNCS*. Springer, Berlin / Heidelberg.

E. A. Issacs and H. H. Clark. 1987. References in conversations between experts and novices. *Journal of Experimental Psychology: General*, 116:26–37.

S. Janarthanam and O. Lemon. 2009a. A Two-tier User Simulation Model for Reinforcement Learning of Adaptive Referring Expression Generation Policies. In *Proc. SigDial'09*.

S. Janarthanam and O. Lemon. 2009b. A Wizard-of-Oz environment to study Referring Expression Generation in a Situated Spoken Dialogue Task. In *Proc. ENLG'09*.

S. Janarthanam and O. Lemon. 2009c. Learning Lexical Alignment Policies for Generating Referring Expressions for Spoken Dialogue Systems. In *Proc. ENLG'09*.

O. Lemon. 2010. Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation. *Computer Speech and Language*. (to appear).

E. Levin, R. Pieraccini, and W. Eckert. 1997. Learning Dialogue Strategies within the Markov Decision Process Framework. In *Proc. of ASRU97*.

K. McKeown, J. Robin, and M. Tanenblatt. 1993. Tailoring Lexical Choice to the User's Vocabulary in Multimedia Explanation Generation. In *Proc. ACL 1993*.

C. L. Paris. 1987. *The Use of Explicit User Models in Text Generations: Tailoring to a User's Level of Expertise*. Ph.D. thesis, Columbia University.

E. Reiter. 1991. Generating Descriptions that Exploit a User's Domain Knowledge. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 257–285. Academic Press.

V. Rieser and O. Lemon. 2009. Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proc. EACL'09*.

V. Rieser and O. Lemon. 2010. Optimising information presentation for spoken dialogue systems. In *Proc. ACL*. (to appear).

J. Schatzmann, K. Weilhammer, M. N. Stuttle, and S. J. Young. 2006. A Survey of Statistical User Simulation Techniques for Reinforcement Learning of Dialogue Management Strategies. *Knowledge Engineering Review*, pages 97–126.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. J. Young. 2007. Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proc of HLT/NAACL 2007*.

D. Shapiro and P. Langley. 2002. Separating skills from preference: Using learning to program by reward. In *Proc. ICML-02*.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press.

A Risk Minimization Framework for Extractive Speech Summarization

Shih-Hsiang Lin and Berlin Chen
National Taiwan Normal University
Taipei, Taiwan
{shlin, berlin}@csie.ntnu.edu.tw

Abstract

In this paper, we formulate extractive summarization as a risk minimization problem and propose a unified probabilistic framework that naturally combines supervised and unsupervised summarization models to inherit their individual merits as well as to overcome their inherent limitations. In addition, the introduction of various loss functions also provides the summarization framework with a flexible but systematic way to render the redundancy and coherence relationships among sentences and between sentences and the whole document, respectively. Experiments on speech summarization show that the methods deduced from our framework are very competitive with existing summarization approaches.

1 Introduction

Automated summarization systems which enable user to quickly digest the important information conveyed by either a single or a cluster of documents are indispensible for managing the rapidly growing amount of textual information and multimedia content (Mani and Maybury, 1999). On the other hand, due to the maturity of text summarization, the research paradigm has been extended to speech summarization over the years (Furui et al., 2004; McKeown et al., 2005). Speech summarization is expected to distill important information and remove redundant and incorrect information caused by recognition errors from spoken documents, enabling user to efficiently review spoken documents and understand the associated topics quickly. It would also be useful for improving the efficiency of a number of potential applications like retrieval and mining of large volumes of spoken documents.

A summary can be either abstractive or extractive. In abstractive summarization, a fluent and concise abstract that reflects the key concepts of a document is generated, whereas in extractive summarization, the summary is usually formed by selecting salient sentences from the original document (Mani and Maybury, 1999). The former requires highly sophisticated natural language processing techniques, including semantic representation and inference, as well as natural language generation, while this would make abstractive approaches difficult to replicate or extend from constrained domains to more general domains. In addition to being extractive or abstractive, a summary may also be generated by considering several other aspects like being generic or query-oriented summarization, single-document or multi-document summarization, and so forth. The readers may refer to (Mani and Maybury, 1999) for a comprehensive overview of automatic text summarization. In this paper, we focus exclusively on generic, single-document extractive summarization which forms the building block for many other summarization tasks.

Aside from traditional ad-hoc extractive summarization methods (Mani and Maybury, 1999), machine-learning approaches with either supervised or unsupervised learning strategies have gained much attention and been applied with empirical success to many summarization tasks (Kupiec et al., 1999; Lin et al., 2009). For supervised learning strategies, the summarization task is usually cast as a two-class (summary and non-summary) sentence-classification problem: A sentence with a set of indicative features is input to the classifier (or summarizer) and a decision is then returned from it on the basis of these features. In general, they usually require a training set, comprised of several documents and their corresponding handcrafted summaries (or labeled data), to train the classifiers. However, manual labeling is expensive in terms of time and personnel. The other potential problem is the so-called "*bag-of-sentences*" assumption implicitly made by most of these summarizers. That is, sentences are classified independently of each other,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 79–87,
Uppsala, Sweden, 11-16 July 2010.

without leveraging the dependence relationships among the sentences or the global structure of the document (Shen et al., 2007).

Another line of thought attempts to conduct document summarization using unsupervised machine-learning approaches, getting around the need for manually labeled training data. Most previous studies conducted along this line have their roots in the concept of sentence *centrality* (Gong and Liu, 2001; Erkan and Radev, 2004; Radev et al., 2004; Mihalcea and Tarau, 2005). Put simply, sentences more similar to others are deemed more salient to the main theme of the document; such sentences thus will be selected as part of the summary. Even though the performance of unsupervised summarizers is usually worse than that of supervised summarizers, their domain-independent and easy-to-implement properties still make them attractive.

Building on these observations, we expect that researches conducted along the above-mentioned two directions could complement each other, and it might be possible to inherit their individual merits to overcome their inherent limitations. In this paper, we present a probabilistic summarization framework stemming from Bayes decision theory (Berger, 1985) for speech summarization. This framework can not only naturally integrate the above-mentioned two modeling paradigms but also provide a flexible yet systematic way to render the redundancy and coherence relationships among sentences and between sentences and the whole document, respectively. Moreover, we also illustrate how the proposed framework can unify several existing summarization models.

The remainder of this paper is structured as follows. We start by reviewing related work on extractive summarization. In Section 3 we formulate the extractive summarization task as a risk minimization problem, followed by a detailed elucidation of the proposed methods in Section 4. Then, the experimental setup and a series of experiments and associated discussions are presented in Sections 5 and 6, respectively. Finally, Section 7 concludes our presentation and discusses avenues for future work.

2 Background

Speech summarization can be conducted using either supervised or unsupervised methods (Furui et al., 2004, McKeown et al., 2005, Lin et al., 2008). In the following, we briefly review a few celebrated methods that have been applied to extractive speech summarization tasks with good success.

2.1 Supervised summarizers

Extractive speech summarization can be treated as a two-class (positive/negative) classification problem. A spoken sentence S_i is characterized by set of T indicative features $X_i = \{x_{i1}, \cdots, x_{iT}\}$, and they may include lexical features (Koumpis and Renals, 2000), structural features (Maskey and Hirschberg, 2003), acoustic features (Inoue et al., 2004), discourse features (Zhang et al., 2007) and relevance features (Lin et al., 2009). Then, the corresponding feature vector X_i of S_i is taken as the input to the classifier. If the output (classification) score belongs to the positive class, S_i will be selected as part of the summary; otherwise, it will be excluded (Kupiec et al., 1999). Specifically, the problem can be formulated as follows: Construct a sentence ranking model that assigns a classification score (or a posterior probability) of being in the summary class to each sentence of a spoken document to be summarized; important sentences are subsequently ranked and selected according to these scores. To this end, several popular machine-learning methods could be utilized, like Bayesian classifier (BC) (Kupiec et al., 1999), Gaussian mixture model (GMM) (Fattah and Ren, 2009) , hidden Markov model (HMM) (Conroy and O'leary, 2001), support vector machine (SVM) (Kolcz et al., 2001), maximum entropy (ME) (Ferrier, 2001), conditional random field (CRF) (Galley, 2006; Shen et al., 2007), to name a few.

Although such supervised summarizers are effective, most of them (except CRF) usually implicitly assume that sentences are independent of each other (the so-called "*bag-of-sentences*" assumption) and classify each sentence individually without leveraging the relationship among the sentences (Shen et al., 2007). Another major shortcoming of these summarizers is that a set of handcrafted document-reference summary exemplars are required for training the summarizers; however, such summarizers tend to limit their generalization capability and might not be readily applicable for new tasks or domains.

2.2 Unsupervised summarizers

The related work conducted along this direction usually relies on some heuristic rules or statistical evidences between each sentence and the document, avoiding the need of manually labeled training data. For example, the vector space model (VSM) approach represents each sentence of a document and the document itself in vector space (Gong and Liu, 2001), and computes the relevance score between each sentence and the document (e.g., the cosine measure of the simi-

larity between two vectors). Then, the sentences with the highest relevance scores are included in the summary. A natural extension is to represent each document or each sentence vector in a latent semantic space (Gong and Liu, 2001), instead of simply using the literal term information as that done by VSM.

On the other hand, the graph-based methods, such as TextRank (Mihalcea and Tarau, 2005) and LexRank (Erkan and Radev, 2004), conceptualize the document to be summarized as a network of sentences, where each node represents a sentence and the associated weight of each link represents the lexical or topical similarity relationship between a pair of nodes. Document summarization thus relies on the global structural information conveyed by such conceptualized network, rather than merely considering the local features of each node (sentence).

However, due to the lack of document-summary reference pairs, the performance of the unsupervised summarizers is usually worse than that of the supervised summarizers. Moreover, most of the unsupervised summarizers are constructed solely on the basis of the lexical information without considering other sources of information cues like discourse features, acoustic features, and so forth.

3 A risk minimization framework for extractive summarization

Extractive summarization can be viewed as a decision making process in which the summarizer attempts to select a representative subset of sentences or paragraphs from the original documents. Among the several analytical methods that can be employed for the decision process, the Bayes decision theory, which quantifies the tradeoff between various decisions and the potential cost that accompanies each decision, is perhaps the most suited one that can be used to guide the summarizer in choosing a course of action in the face of some uncertainties underlying the decision process (Berger, 1985). Stated formally, a decision problem may consist of four basic elements: 1) an observation O from a random variable $\mathbf{O}$, 2) a set of possible decisions (or actions) $a \in \mathbf{A}$, 3) the state of nature $\theta \in \mathbf{\Theta}$, and 4) a loss function $L(a_i, \theta)$ which specifies the cost associated with a chosen decision a_i given that θ is the true state of nature. The expected risk (or conditional risk) associated with taking decision a_i is given by

$$R(a_i \mid O) = \int_\theta L(a_i, \theta) p(\theta \mid O) d\theta, \quad (1)$$

where $p(\theta|O)$ is the posterior probability of the state of nature being θ given the observation O. Bayes decision theory states that the optimum decision can be made by contemplating each action a_i, and then choosing the action for which the expected risk is minimum:

$$a^* = \arg\min_{a_i} R(a_i \mid O). \quad (2)$$

The notion of minimizing the Bayes risk has gained much attention and been applied with success to many natural language processing (NLP) tasks, such as automatic speech recognition (Goel and Byrne, 2000), statistical machine translation (Kumar and Byrne, 2004) and statistical information retrieval (Zhai and Lafferty, 2006). Following the same spirit, we formulate the extractive summarization task as a Bayes risk minimization problem. Without loss of generality, let us denote $\pi \in \mathbf{\Pi}$ as one of possible selection strategies (or state of nature) which comprises a set of indicators used to address the importance of each sentence S_i in a document D to be summarized. A feasible selection strategy can be fairly arbitrary according to the underlying principle. For example, it could be a set of binary indicators denoting whether a sentence should be selected as part of summary or not. On the contrary, it may also be a ranked list used to address the significance of each individual sentence. Moreover, we refer to the k-th action a_k as choosing the k-th selection strategy π_k, and the observation O as the document D to be summarized. As a result, the expected risk of a certain selection strategy π_k is given by

$$R(\pi_k \mid D) = \int_\pi L(\pi_k, \pi) p(\pi \mid D) d\pi. \quad (3)$$

Consequently, the ultimate goal of extractive summarization could be stated as the search of the best selection strategy from the space of all possible selection strategies that minimizes the expected risk defined as follows:

$$\begin{aligned} \pi^* &= \arg\min_{\pi_k} R(\pi_k \mid D) \\ &= \arg\min_{\pi_k} \int_\pi L(\pi_k, \pi) p(\pi \mid D) d\pi. \end{aligned} \quad (4)$$

Although we have described a general formulation for the extractive summarization problem on the grounds of the Bayes decision theory, we consider hereafter a special case of it where the selection strategy is represented by a binary decision vector, of which each element corresponds to a specific sentence S_i in the document D and designates whether it should be selected as part of the summary or not, as the first such attempt. More concretely, we assume that the summary

sentences of a given document can be iteratively chosen (i.e., one at each iteration) from the document until the aggregated summary reaches a predefined target summarization ratio. It turns out that the binary vector for each possible action will have just one element equal to 1 and all others equal to zero (or the so-called "*one-of-n*" coding). For ease of notation, we denote the binary vector by S_i when the i-th element has a value of 1. Therefore, the risk minimization framework can be reduced to

$$\begin{aligned} S^* &= \arg\min_{S_i \in \widetilde{D}} R(S_i \mid \widetilde{D}) \\ &= \arg\min_{S_i \in \widetilde{D}} \sum_{S_j \in \widetilde{D}} L(S_i, S_j) P(S_j \mid \widetilde{D}), \end{aligned} \tag{5}$$

where $\widetilde{D}$ denotes the remaining sentences that have not been selected into the summary yet (i.e., the "*residual*" document); $P(S_j \mid \widetilde{D})$ is the posterior probability of a sentence S_j given $\widetilde{D}$. According to the Bayes' rule, we can further express $P(S_j \mid \widetilde{D})$ as (Chen et al., 2009)

$$P(S_j \mid \widetilde{D}) = \frac{P(\widetilde{D} \mid S_j) P(S_j)}{P(\widetilde{D})}, \tag{6}$$

where $P(\widetilde{D} \mid S_j)$ is the sentence generative probability, i.e., the likelihood of $\widetilde{D}$ being generated by S_j; $P(S_j)$ is the prior probability of S_j being important; and the evidence $P(\widetilde{D})$ is the marginal probability of $\widetilde{D}$, which can be approximated by

$$P(\widetilde{D}) \approx \sum_{S_m \in \widetilde{D}} P(\widetilde{D} \mid S_m) P(S_m). \tag{7}$$

By substituting (6) and (7) into (5), we obtain the following final selection strategy for extractive summarization:

$$S^* = \arg\min_{S_i \in \widetilde{D}} \sum_{S_j \in \widetilde{D}} L(S_i, S_j) \frac{P(\widetilde{D} \mid S_j) P(S_j)}{\sum_{S_m \in \widetilde{D}} P(\widetilde{D} \mid S_m) P(S_m)}. \tag{8}$$

A remarkable feature of this framework lies in that a sentence to be considered as part of the summary is actually evaluated by three different fundamental factors: (1) $P(S_j)$ is the sentence prior probability that addresses the importance of sentence S_j itself; (2) $P(\widetilde{D} \mid S_j)$ is the sentence generative probability that captures the degree of relevance of S_j to the residual document $\widetilde{D}$; and (3) $L(S_i, S_j)$ is the loss function that characterizes the relationship between sentence S_i and any other sentence S_j. As we will soon see, such a framework can be regarded as a generalization of several existing summarization methods. A detailed account on the construction of these three component models in the framework will be given in the following section.

4 Proposed Methods

There are many ways to construct the above mentioned three component models, i.e., the sentence generative model $P(\widetilde{D} \mid S_j)$, the sentence prior model $P(S_j)$, and the loss function $L(S_i, S_j)$. In what follows, we will shed light on one possible attempt that can accomplish this goal elegantly.

4.1 Sentence generative model

In order to estimate the sentence generative probability, we explore the language modeling (LM) approach, which has been introduced to a wide spectrum of IR tasks and demonstrated with good empirical success, to predict the sentence generative probability. In the LM approach, each sentence in a document can be simply regarded as a probabilistic generative model consisting of a unigram distribution (the so-called "*bag-of-words*" assumption) for generating the document (Chen et al., 2009):

$$P(\widetilde{D} \mid S_j) = \prod_{w \in \widetilde{D}} P(w \mid S_j)^{c(w, \widetilde{D})}, \tag{9}$$

where $c(w, \widetilde{D})$ is the number of times that index term (or word) w occurs in $\widetilde{D}$, reflecting that w will contribute more in the calculation of $P(\widetilde{D} \mid S_j)$ if it occurs more frequently in $\widetilde{D}$. Note that the sentence model $P(w \mid S_j)$ is simply estimated on the basis of the frequency of index term w occurring in the sentence S_j with the maximum likelihood (ML) criterion. In a sense, (9) belongs to a kind of literal term matching strategy (Chen, 2009) and may suffer the problem of unreliable model estimation owing particularly to only a few sampled index terms present in the sentence (Zhai, 2008). To mitigate this potential defect, a unigram probability estimated from a general collection, which models the general distribution of words in the target language, is often used to smooth the sentence model. Interested readers may refer to (Zhai, 2008; Chen et al., 2009) for a thorough discussion on various ways to construct the sentence generative model.

4.2 Sentence prior model

The sentence prior probability $P(S_j)$ can be regarded as the likelihood of a sentence being important without seeing the whole document. It could be assumed uniformly distributed over sentences or estimated from a wide variety of factors, such as the lexical information, the structural information or the inherent prosodic properties of a spoken sentence.

A straightforward way is to assume that the sentence prior probability $P(S_j)$ is in proportion to the posterior probability of a sentence S_j be-

ing included in the summary class when observing a set of indicative features X_j of S_j derived from such factors or other sentence importance measures (Kupiec et al., 1999). These features can be integrated in a systematic way into the proposed framework by taking the advantage of the learning capability of the supervised machine-learning methods. Specifically, the prior probability $P(S_j)$ can be approximated by:

$$P(S_j) \approx \frac{p(X_j \mid \mathbf{S})P(\mathbf{S})}{P(X_j \mid \mathbf{S})P(\mathbf{S}) + P(X_j \mid \overline{\mathbf{S}})P(\overline{\mathbf{S}})}, \quad (10)$$

where $P(X_j \mid \mathbf{S})$ and $P(X_j \mid \overline{\mathbf{S}})$ are the likelihoods that a sentence S_j with features X_j are generated by the summary class $\mathbf{S}$ and the non-summary class $\overline{\mathbf{S}}$, respectively; the prior probability $P(\mathbf{S})$ and $P(\overline{\mathbf{S}})$ are set to be equal in this research. To estimate $P(X_j \mid \mathbf{S})$ and $P(X_j \mid \overline{\mathbf{S}})$, several popular supervised classifiers (or summarizers), like BC or SVM, can be leveraged for this purpose.

4.3 Loss function

The loss function introduced in the proposed summarization framework is to measure the relationship between any pair of sentences. Intuitively, when a given sentence is more dissimilar from most of the other sentences, it may incur higher loss as it is taken as the representative sentence (or summary sentence) to represent the main theme embedded in the other ones. Consequently, the loss function can be built on the notion of the similarity measure. In this research, we adopt the cosine measure (Gong and Liu, 2001) to fulfill this goal. We first represent each sentence S_i in vector form where each dimension specifies the weighted statistic $z_{t,i}$, e.g., the product of the term frequency (TF) and inverse document frequency (IDF) scores, associated with an index term w_t in sentence S_i. Then, the cosine similarity between any given two sentences (S_i, S_j) is

$$Sim(S_i, S_j) = \frac{\sum_{t=1}^{T} z_{t,i} \times z_{t,j}}{\sqrt{\sum_{t=1}^{T} z_{t,i}^2} \times \sqrt{\sum_{t=1}^{T} z_{t,j}^2}}. \quad (10)$$

The loss function is thus defined by

$$L(S_i, S_j) = 1 - Sim(S_i, S_j). \quad (11)$$

Once the sentence generative model $P(\widetilde{D} \mid S_j)$, the sentence prior model $P(S_j)$ and the loss function $L(S_i, S_j)$ have been properly estimated, the summary sentences can be selected iteratively by (8) according to a predefined target summarization ratio. However, as can be seen from (8), a new summary sentence is selected without considering the redundant information that is also contained in the already selected summary sentences. To alleviate this problem, the concept of maximum marginal relevance (MMR) (Carbonell and Goldstein, 1998), which performs sentence selection iteratively by striking the balance between topic relevance and coverage, can be incorporated into the loss function:

$$L(S_i, S_j) = 1 - \left[\begin{array}{l} \beta \cdot Sim(S_i, S_j) \\ -(1-\beta) \cdot \max_{S' \in \mathbf{Summ}} Sim(S_i, S') \end{array} \right], \quad (12)$$

where **Summ** represents the set of sentences that have already been included into the summary and the novelty factor β is used to trade off between relevance and redundancy.

4.4 Relation to other summarization models

In this subsection, we briefly illustrate the relationship between our proposed summarization framework and a few existing summarization approaches. We start by considering a special case where a 0-1 loss function is used in (8), namely, the loss function will take value 0 if the two sentences are identical, and 1 otherwise. Then, (8) can be alternatively represented by

$$\begin{aligned} S^* &= \arg\min_{S_i \in \widetilde{D}} \sum_{S_j \in \widetilde{D}, S_j \neq S_i} \frac{P(\widetilde{D} \mid S_j)P(S_j)}{\sum_{S_m \in \widetilde{D}} P(\widetilde{D} \mid S_m)P(S_m)} \\ &= \arg\max_{S_i \in \widetilde{D}} \frac{P(\widetilde{D} \mid S_i)P(S_i)}{\sum_{S_m \in \widetilde{D}} P(\widetilde{D} \mid S_m)P(S_m)}, \end{aligned} \quad (13)$$

which actually provides a natural integration of the supervised and unsupervised summarizers (Lin et al., 2009), as mentioned previously.

If we further assume the prior probability $P(S_j)$ is uniformly distributed, the important (or summary) sentence selection problem has now been reduced to the problem of measuring the document-likelihood $P(\widetilde{D} \mid S_j)$, or the relevance between the document and the sentence. Alone a similar vein, the important sentences of a document can be selected (or ranked) solely based on the prior probability $P(S_j)$ with the assumption of an equal document-likelihood $P(\widetilde{D} \mid S_j)$.

5 Experimental setup

5.1 Data

The summarization dataset used in this research is a widely used broadcast news corpus collected by the Academia Sinica and the Public Television Service Foundation of Taiwan between November 2001 and April 2003 (Wang et al., 2005). Each story contains the speech of one studio anchor, as well as several field reporters and interviewees. A subset of 205 broadcast news doc-

Kappa	ROGUE-1	ROUGE-2	ROUGE-L
0.400	0.600	0.532	0.527

Table 1: The agreement among the subjects for important sentence ranking for the evaluation set.

Structural features	1.Duration of the current sentence 2.Position of the current sentence 3.Length of the current sentence
Lexical Features	1.Number of named entities 2.Number of stop words 3.Bigram language model scores 4.Normalized bigram scores
Acoustic Features	1.The 1st formant 2.The 2nd formant 3.The pitch value 4.The peak normalized cross-correlation of pitch
Relevance Feature	1.VSM score

Table 2: Basic sentence features used by BC.

uments compiled between November 2001 and August 2002 was reserved for the summarization experiments.

Three subjects were asked to create summaries of the 205 spoken documents for the summarization experiments as references (the gold standard) for evaluation. The summaries were generated by ranking the sentences in the reference transcript of a spoken document by importance without assigning a score to each sentence. The average Chinese character error rate (CER) obtained for the 205 spoken documents was about 35%.

Since broadcast news stories often follow a relatively regular structure as compared to other speech materials like conversations, the positional information would play an important (dominant) role in extractive summarization of broadcast news stories; we, hence, chose 20 documents for which the generation of reference summaries is less correlated with the positional information (or the position of sentences) as the held-out test set to evaluate the general performance of the proposed summarization framework, and 100 documents as the development set.

5.2 Performance evaluation

For the assessment of summarization performance, we adopted the widely used ROUGE measure (Lin, 2004) because of its higher correlation with human judgments. It evaluates the quality of the summarization by counting the number of overlapping units, such as *N*-grams, longest common subsequences or skip-bigram, between the automatic summary and a set of reference summaries. Three variants of the ROGUE measure were used to quantify the utility of the proposed method. They are, respectively, the ROUGE-1 (unigram) measure, the ROUGE-2 (bigram) measure and the ROUGE-L (longest common subsequence) measure (Lin, 2004).

The summarization ratio, defined as the ratio of the number of words in the automatic (or manual) summary to that in the reference transcript of a spoken document, was set to 10% in this research. Since increasing the summary length tends to increase the chance of getting higher scores in the recall rate of the various ROUGE measures and might not always select the right number of informative words in the automatic summary as compared to the reference summary, all the experimental results reported hereafter are obtained by calculating the F-scores of these ROUGE measures, respectively (Lin, 2004). Table 1 shows the levels of agreement (the Kappa statistic and ROUGE measures) between the three subjects for important sentence ranking. They seem to reflect the fact that people may not always agree with each other in selecting the important sentences for representing a given document.

5.3 Features for supervised summarizers

We take BC as the representative supervised summarizer to study in this paper. The input to BC consists of a set of 28 indicative features used to characterize a spoken sentence, including the structural features, the lexical features, the acoustic features and the relevance feature. For each kind of acoustic features, the minimum, maximum, mean, difference value and mean difference value of a spoken sentence are extracted. The difference value is defined as the difference between the minimum and maximum values of the spoken sentence, while the mean difference value is defined as the mean difference between a sentence and its previous sentence. Finally, the relevance feature (VSM score) is use to measure the degree of relevance for a sentence to the whole document (Gong and Liu, 2001). These features are outlined in Table 2, where each of them was further normalized to zero mean and unit variance.

6 Experimental results and discussions

6.1 Baseline experiments

In the first set of experiments, we evaluate the baseline performance of the LM and BC summarizers (cf. Sections 4.1 and 4.2), respectively. The corresponding results are detailed in Table 3,

	Text Document (TD)			Spoken Document (SD)		
	ROGUE-1	ROUGE-2	ROUGE-L	ROGUE-1	ROUGE-2	ROUGE-L
BC	0.445 (0.390 - 0.504)	0.346 (0.201 - 0.415)	0.404 (0.348 - 0.468)	0.369 (0.316 - 0.426)	0.241 (0.183 - 0.302)	0.321 (0.268 - 0.378)
LM	0.387 (0.302 - 0.474)	0.264 (0.168 - 0.366)	0.334 (0.251 - 0.415)	0.319 (0.274 - 0.367)	0.164 (0.115 - 0.224)	0.253 (0.215 - 0.301)

Table 3: The results achieved by the BC and LM summarizers, respectively.

		Text Document (TD)			Spoken Document (SD)		
Prior	Loss	ROGUE-1	ROUGE-2	ROUGE-L	ROGUE-1	ROUGE-2	ROUGE-L
BC	0-1	0.501	0.401	0.459	0.417	0.281	0.356
	SIM	0.524	0.425	0.473	0.475	0.351	0.420
	MMR	0.529	0.426	0.479	0.475	0.351	0.420
Uniform	SIM	0.405	0.281	0.348	0.365	0.209	0.305
	MMR	0.417	0.282	0.359	0.391	0.236	0.338

Table 4: The results achieved by several methods derived from the proposed summarization framework.

where the values in the parentheses are the associated 95% confidence intervals. It is also worth mentioning that TD denotes the summarization results obtained based on manual transcripts of the spoken documents while SD denotes the results using the speech recognition transcripts which may contain speech recognition errors and sentence boundary detection errors. In this research, sentence boundaries were determined by speech pauses. For the TD case, the acoustic features were obtained by aligning the manual transcripts to their spoken documents counterpart by performing word-level forced alignment.

Furthermore, the ROGUE measures, in essence, are evaluated by counting the number of overlapping units between the automatic summary and the reference summary; the corresponding evaluation results, therefore, would be severely affected by speech recognition errors when applying the various ROUGE measures to quantify the performance of speech summarization. In order to get rid of the cofounding effect of this factor, it is assumed that the selected summary sentences can also be presented in speech form (besides text form) such that users can directly listen to the audio segments of the summary sentences to bypass the problem caused by speech recognition errors. Consequently, we can align the ASR transcripts of the summary sentences to their respective audio segments to obtain the correct (manual) transcripts for the summarization performance evaluation (i.e., for the SD case).

Observing Table 3 we notice two particularities. First, there are significant performance gaps between summarization using the manual transcripts and the erroneous speech recognition transcripts. The relative performance degradations are about 15%, 34% and 23%, respectively, for ROUGE-1, ROUGE2 and ROUGE-L measures. One possible explanation is that the erroneous speech recognition transcripts of spoken sentences would probably carry wrong information and thus deviate somewhat from representing the true theme of the spoken document. Second, the supervised summarizer (i.e., BC) outperforms the unsupervised summarizer (i.e., LM). The better performance of BC can be further explained by two reasons. One is that BC is trained with the handcrafted document-summary sentence labels in the development set while LM is instead conducted in a purely unsupervised manner. Another is that BC utilizes a rich set of features to characterize a given spoken sentence while LM is constructed solely on the basis of the lexical (unigram) information.

6.2 Experiments on the proposed methods

We then turn our attention to investigate the utility of several methods deduced from our proposed summarization framework. We first consider the case when a 0-1 loss function is used (cf. (13)), which just show a simple combination of BC and LM. As can be seen from the first row of Table 4, such a combination can give about 4% to 5% absolute improvements as compared to the results of BC illustrated in Table 3. It in some sense confirms the feasibility of combining the supervised and unsupervised summarizers. Moreover, we consider the use of the loss functions defined in (11) (denoted by SIM) and (12) (denoted by MMR), and the corresponding results are shown in the second and the third rows of Table 4, respectively. It can be found that

MMR delivers higher summarization performance than SIM (especially for the SD case), which in turn verifies the merit of incorporating the MMR concept into the proposed framework for extractive summarization. If we further compare the results achieved by MMR with those of BC and LM as shown in Table 3, we can find significant improvements both for the TD and SD cases. By and large, for the TD case, the proposed summarization method offers relative performance improvements of about 19%, 23% and 19%, respectively, in the ROUGE-1, ROUGE-2 and ROUGE-L measures as compared to the BC baseline; while the relative improvements are 29%, 46% and 31%, respectively, in the same measurements for the SD case. On the other hand, the performance gap between the TD and SD cases are reduced to a good extent by using the proposed summarization framework.

In the next set of experiments, we simply assume the sentence prior probability $P(S_j)$ defined in (8) is uniformly distributed, namely, we do not use any supervised information cue but use the lexical information only. The importance of a given sentence is thus considered from two angles: 1) the relationship between a sentence and the whole document, and 2) the relationship between the sentence and the other individual sentences. The corresponding results are illustrated in the lower part of Table 4 (denoted by Uniform). We can see that the additional consideration of the sentence-sentence relationship appears to be beneficial as compared to that only considering the document-sentence relevance information (cf. the second row of Table 3). It also gives competitive results as compared to the performance of BC (cf. the first row of Table 3) for the SD case.

6.3 Comparison with conventional summarization methods

In the final set of experiments, we compare our proposed summarization methods with a few existing summarization methods that have been widely used in various summarization tasks, including LEAD, VSM, LexRank and CRF; the corresponding results are shown in Table 5. It should be noted that the LEAD-based method simply extracts the first few sentences in a document as the summary. To our surprise, CRF does not provide superior results as compared to the other summarization methods. One possible explanation is that the structural evidence of the spoken documents in the test set is not strong enough for CRF to show its advantage of modeling the local structural information among sentences. On the other hand, LexRank gives a very promising performance in spite that it only utilizes lexical information in an unsupervised manner. This somewhat reflects the importance of capturing the global relationship for the sentences in the spoken document to be summarized. As compared to the results shown in the "BC" part of Table 4, we can see that our proposed methods significantly outperform all the conventional summarization methods compared in this paper, especially for the SD case.

		ROGUE-1	ROUGE-2	ROUGE-L
LEAD	TD	0.320	0.197	0.283
	SD	0.312	0.168	0.251
VSM	TD	0.345	0.220	0.287
	SD	0.337	0.189	0.277
LexRank	TD	0.435	0.314	0.377
	SD	0.348	0.204	0.294
CRF	TD	0.431	0.315	0.383
	SD	0.358	0.220	0.291

Table 5: The results achieved by four conventional summarization methods.

7 Conclusions and future work

We have proposed a risk minimization framework for extractive speech summarization, which enjoys several advantages. We have also presented a simple yet effective implementation that selects the summary sentences in an iterative manner. Experimental results demonstrate that the methods deduced from such a framework can yield substantial improvements over several popular summarization methods compared in this paper. We list below some possible future extensions: 1) integrating different selection strategies, e.g., the listwise strategy that defines the loss function on all the sentences associated with a document to be summarized, into this framework, 2) exploring different modeling approaches for this framework, 3) investigating discriminative training criteria for training the component models in this framework, and 4) extending and applying the proposed framework to multi-document summarization tasks.

References

James O. Berger *Statistical decision theory and Bayesian analysis*. Springer-Verlap, 1985.

Berlin Chen. 2009. Word topic models for spoken document retrieval and transcription. *ACM Transactions on Asian Language Information Processing*, 8, (1): 2:1 - 2:27.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. of Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval: 335 - 336.

Yi-Ting Chen, Berlin Chen and Hsin-Min Wang. 2009. A probabilistic generative framework for extractive broadcast news speech summarization. *IEEE Transactions on Audio, Speech and Language Processing*, 17, (1): 95 - 106.

John M. Conroy and Dianne P. O'Leary. 2001. Text summarization via hidden Markov models. In *Proc. of Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: 406 - 407.

Güneş Erkan and Dragomir R. Radev. 2004. LexRank: graph-based lexical centrality as salience in text summarization. *Journal or Artificial Intelligence Research*, 22: 457 - 479.

Mohamed Abdel Fattah and Fuji Ren. 2009. GA, MR, FFNN, PNN and GMM based models for automatic text summarization. *Computer Speech and Language*, 23, (1): 126 - 144.

Louisa Ferrier *A maximum entropy approach to text summarization*. School of Artificial Intelligence, University of Edinburgh, 2001.

Sadaoki Furui, Tomonori Kikuchi, Yousuke Shinnaka and Chiori Hori. 2004. Speech-to-text and speech-to-speech summarization of spontaneous speech. *IEEE Transactions on Speech and Audio Processing*, 12, (4): 401 - 408.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proc. of Conference on Empirical Methods in Natural Language Processing*: 364 - 372.

Vaibhava Goel and William Byrne. 2000. Minimum Bayes-risk automatic speech recognition. *Computer Speech and Language*, 14, (2): 115 - 135.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proc. of Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: 19 - 25.

Akira Inoue, Takayoshi Mikami and Yoichi Yamashita. 2004. Improvement of speech summarization using prosodic information, In *Proc. of Speech Prosody*: 599 - 602.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proc. of Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*: 169 - 176.

Aleksander Kolcz, Vidya Prabakarmurthi and Jugal Kalita. 2001. Summarization as feature selection for text categorization. In *Proc. of Conference on Information and Knowledge Management*: 365 - 370.

Julian Kupiec, Jan Pedersen and Francine Chen. 1999. A trainable document summarizer. In *Proc. of Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: 68 - 73.

Konstantinos Koumpis and Steve Renals. 2000. Transcription And Summarization Of Voicemail Speech. In *Proc. of International Conference on Spoken Language Processing*: 688 - 691.

Chin-Yew Lin. 2004. ROUGE: a Package for Automatic Evaluation of Summaries. In *Proc. of Workshop on Text Summarization Branches Out*.

Shih-Hsiang Lin, Berlin Chen and Hsin-Min Wang. 2009. A comparative study of probabilistic ranking models for Chinese spoken document summarization. *ACM Transactions on Asian Language Information Processing*, 8, (1): 3:1 - 3:23.

Shih-Hsiang Lin, Yueng-Tien Lo, Yao-Ming Yeh and Berlin Chen. 2009. Hybrids of supervised and unsupervised models for extractive speech summarization. In *Proc. of Annual Conference of the International Speech Communication Association*: 1507 - 1510.

Inderjeet Mani and Mark T. Maybury *Advances in automatic text summarization*. MIT Press, Cambridge, 1999.

Sameer R. Maskey and Julia Hirschberg. 2003. Automatic Summarization of Broadcast News using Structural Features. In Proc. of *the European Conf. Speech Communication and Technology*: 1173 - 1176.

Kathleen McKeown, Julia Hirschberg, Michel Galley and Sameer Maskey. 2005. From text to speech summarization. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*: 997 - 1000.

Rada Mihalcea and Paul Tarau. 2005. TextRank: bringing order into texts. In *Proc. of Conference on Empirical Methods in Natural Language Processing*: 404 - 411.

Dragomir R. Radev, Hongyan Jing, Małgorzata Stys and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919 - 938.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang and Zheng Chen. 2007. Document summarization using conditional random fields. In *Proc. of International Joint Conference on Artificial Intelligence*: 2862 - 2867.

Hsin-Min Wang, Berlin Chen, Jen-Wei Kuo and Shih-Sian Cheng. 2005. MATBN: A Mandarin Chinese broadcast news corpus. *International Journal of Computational Linguistics and Chinese Language Processing*, 10, (2): 219 - 236.

ChengXiang Zhai and John Lafferty. 2006. A risk minimization framework for information retrieval. *Information Processing & Management*, 42, (1): 31 - 55.

ChengXiang Zhai. *Statistical language models for information retrieval*. Morgan & Claypool Publishers, 2008.

Justin Jian Zhang, Ho Yin Chan and Pascale Fung. 2007. Improving Lecture Speech Summarization Using Rhetorical Information. In *Proc. of Workshop of Automatic Speech Recognition Understanding*: 195 - 200.

The Human Language Project: Building a Universal Corpus of the World's Languages

Steven Abney
University of Michigan
abney@umich.edu

Steven Bird
University of Melbourne and
University of Pennsylvania
sbird@unimelb.edu.au

Abstract

We present a grand challenge to build a corpus that will include all of the world's languages, in a consistent structure that permits large-scale cross-linguistic processing, enabling the study of universal linguistics. The focal data types, bilingual texts and lexicons, relate each language to one of a set of reference languages. We propose that the ability to train systems to translate into and out of a given language be the yardstick for determining when we have successfully captured a language. We call on the computational linguistics community to begin work on this Universal Corpus, pursuing the many strands of activity described here, as their contribution to the global effort to document the world's linguistic heritage before more languages fall silent.

1 Introduction

The grand aim of linguistics is the construction of a universal theory of human language. To a computational linguist, it seems obvious that the first step is to collect significant amounts of primary data for a large variety of languages. Ideally, we would like a complete digitization of every human language: a Universal Corpus.

If we are ever to construct such a corpus, it must be now. With the current rate of language loss, we have only a small window of opportunity before the data is gone forever. Linguistics may be unique among the sciences in the crisis it faces. The next generation will forgive us for the most egregious shortcomings in theory construction and technology development, but they will not forgive us if we fail to preserve vanishing primary language data in a form that enables future research.

The scope of the task is enormous. At present, we have non-negligible quantities of machine-readable data for only about 20–30 of the world's 6,900 languages (Maxwell and Hughes, 2006). Linguistics as a field is awake to the crisis. There has been a tremendous upsurge of interest in documentary linguistics, the field concerned with the the "creation, annotation, preservation, and dissemination of transparent records of a language" (Woodbury, 2010). However, documentary linguistics alone is not equal to the task. For example, no million-word machine-readable corpus exists for any endangered language, even though such a quantity would be necessary for wide-ranging investigation of the language once no speakers are available. The chances of constructing large-scale resources will be greatly improved if computational linguists contribute their expertise.

This collaboration between linguists and computational linguists will extend beyond the construction of the Universal Corpus to its exploitation for both theoretical and technological ends. We envisage a new paradigm of universal linguistics, in which grammars of individual languages are built from the ground up, combining expert manual effort with the power tools of probabilistic language models and grammatical inference. A universal grammar captures redundancies which exist across languages, constituting a "universal linguistic prior," and enabling us to identify the distinctive properties of specific languages and families. The linguistic prior and regularities due to common descent enable a new economy of scale for technology development: cross-linguistic triangulation can improve performance while reducing per-language data requirements.

Our aim in the present paper is to move beyond generalities to a concrete plan of attack, and to challenge the field to a communal effort to create a Universal Corpus of the world's languages, in consistent machine-readable format, permitting large-scale cross-linguistic processing.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 88–97,
Uppsala, Sweden, 11-16 July 2010.

2 Human Language Project

2.1 Aims and scope

Although language endangerment provides urgency, the corpus is not intended primarily as a Noah's Ark for languages. The aims go beyond the current crisis: we wish to support cross-linguistic research and technology development at the largest scale. There are existing collections that contain multiple languages, but it is rare to have consistent formats and annotation across languages, and few such datasets contain more than a dozen or so languages.

If we think of a multi-lingual corpus as consisting of an array of items, with columns representing languages and rows representing resource types, the usual focus is on "vertical" processing. Our particular concern, by contrast, is "horizontal" processing that cuts indiscriminately across languages. Hence we require an unusual degree of consistency across languages.

The kind of processing we wish to enable is much like the large-scale systematic research that motivated the Human Genome Project.

> One of the greatest impacts of having the sequence may well be in enabling an entirely new approach to biological research. In the past, researchers studied one or a few genes at a time. With whole-genome sequences ... they can approach questions systematically and on a grand scale. They can study ... how tens of thousands of genes and proteins work together in interconnected networks to orchestrate the chemistry of life. (Human Genome Project, 2007)

We wish to make it possible to investigate human language equally systematically and on an equally grand scale: a Human Linguome Project, as it were, though we have chosen the "Human Language Project" as a more inviting title for the undertaking. The product is a Universal Corpus,[1] in two senses of *universal:* in the sense of including (ultimately) all the world's languages, and in the sense of enabling software and processing methods that are language-universal.

However, we do *not* aim for a collection that is universal in the sense of encompassing all language documentation efforts. Our goal is the construction of a specific resource, albeit a very large resource. We contrast the proposed effort with general efforts to develop open resources, standards, and best practices. We do *not* aim to be all-inclusive. The project does require large-scale collaboration, and a task definition that is simple and compelling enough to achieve buy-in from a large number of data providers. But we do not need and do not attempt to create consensus across the entire community. (Although one can hope that what proves successful for a project of this scale will provide a good foundation for future standards.)

Moreover, we do not aim to collect data merely in the vague hope that it will prove useful. Although we strive for maximum generality, we also propose a specific driving "use case," namely, machine translation (MT), (Hutchins and Somers, 1992; Koehn, 2010). The corpus provides a testing ground for the development of MT system-construction methods that are dramatically "leaner" in their resource requirements, and which take advantage of cross-linguistic bootstrapping. The large engineering question is how one can turn the size of the task—constructing MT systems for all the world's languages simultaneously—to one's advantage, and thereby consume dramatically less data per language.

The choice of MT as the use case is also driven by scientific considerations. To explain, we require a bit of preamble.

We aim for a *digitization* of each human language. What exactly does it mean to digitize an entire language? It is natural to think in terms of replicating the body of resources available for well-documented languages, and the pre-eminent resource for any language is a treebank. Producing a treebank involves a staggering amount of manual effort. It is also notoriously difficult to obtain agreement about how parse trees should be defined in one language, much less in many languages simultaneously. The idea of producing treebanks for 6,900 languages is quixotic, to put it mildly. But is a treebank actually necessary?

Let us suppose that the purpose of a parse tree is to mediate interpretation. A treebank, arguably, represents a theoretical hypothesis about how interpretations could be constructed; the primary data is actually the interpretations themselves. This suggests that we annotate sentences with representations of meanings instead of syntactic structures. Now that seems to take us out of the frying pan into the fire. If obtaining consen-

[1] http://universalcorpus.org/

sus on parse trees is difficult, obtaining consensus on meaning representations is impossible. However, if the language under consideration is anything other than English, then a translation into English (or some other reference language) is for most purposes a perfectly adequate meaning representation. That is, we view machine translation as an approximation to language understanding.

Here is another way to put it. One measure of adequacy of a language digitization is the ability of a human—already fluent in a reference language—to acquire fluency in the digitized language using only archived material. Now it would be even better if we could use a language digitization to construct an *artificial speaker* of the language. Importantly, we do not need to solve the AI problem: the speaker need not decide *what* to say, only how to translate from meanings to sentences of the language, and from sentences back to meanings. Taking sentences in a reference language as the meaning representation, we arrive back at machine translation as the measure of success. *In short, we have successfully captured a language if we can translate into and out of the language.*

The key resource that should be built for each language, then, is a collection of primary texts with translations into a reference language. "Primary text" includes both written documents and transcriptions of recordings. Large volumes of primary texts will be useful even without translation for such tasks as language modeling and unsupervised learning of morphology. Thus, we anticipate that the corpus will have the usual "pyramidal" structure, starting from a base layer of unannotated text, some portion of which is translated into a reference language at the document level to make the next layer. Note that, for maximally authentic primary texts, we assume the direction of translation will normally be from primary text to reference language, not the other way around.

Another layer of the corpus consists of sentence and word alignments, required for training and evaluating machine translation systems, and for extracting bilingual lexicons. Curating such annotations is a more specialized task than translation, and so we expect it will only be done for a subset of the translated texts.

In the last and smallest layer, morphology is annotated. This supports the development of morphological analyzers, to preprocess primary texts to identify morpheme boundaries and recognize allomorphs, reducing the amount of data required for training an MT system. This most-refined target annotation corresponds to the *interlinear glossed texts* that are the de facto standard of annotation in the documentary linguistics community.

We postulate that interlinear glossed text is sufficiently fine-grained to serve our purposes. It invites efforts to enrich it by automatic means: for example, there has been work on parsing the English translations and using the word-by-word glosses to transfer the parse tree to the object language, effectively creating a treebank automatically (Xia and Lewis, 2007). At the same time, we believe that interlinear glossed text is sufficiently simple and well-understood to allow rapid construction of resources, and to make cross-linguistic consistency a realistic goal.

Each of these layers—primary text, translations, alignments, and morphological glosses—seems to be an unavoidable piece of the overall solution. The fact that these layers will exist in diminishing quantity is also unavoidable. However, there is an important consequence: the primary texts will be permanently subject to new translation initiatives, which themselves will be subject to new alignment and glossing initiatives, in which each step is an instance of semisupervised learning (Abney, 2007). As time passes, our ability to enhance the quantity and quality of the annotations will only increase, thanks to effective combinations of automatic, professional, and crowd-sourced effort.

2.2 Principles

The basic principles upon which the envisioned corpus is based are the following:

Universality. Covering as many languages as possible is the first priority. Progress will be gauged against concrete goals for numbers of languages, data per language, and coverage of language families (Whalen and Simons, 2009).

Machine readability and consistency. "Covering" languages means enabling machine processing seamlessly across languages. This will support new types of linguistic inquiry and the development and testing of inference methods (for morphology, parsers, machine translation) across large numbers of typologically diverse languages.

Community effort. We cannot expect a single organization to assemble a resource on this scale. It will be necessary to get community buy-in, and

many motivated volunteers. The repository will not be the sole possession of any one institution.

Availability. The content of the corpus will be available under one or more permissive licenses, such as the Creative Commons Attribution License (CC-BY), placing as few limits as possible on community members' ability to obtain and enhance the corpus, and redistribute derivative data.

Utility. The corpus aims to be maximally useful, and minimally parochial. Annotation will be as lightweight as possible; richer annotations will will emerge bottom-up as they prove their utility at the large scale.

Centrality of primary data. Primary texts and recordings are paramount. Secondary resources such as grammars and lexicons are important, but no substitute for primary data. It is desirable that secondary resources be integrated with—if not derived from—primary data in the corpus.

2.3 What to include

What should be included in the corpus? To some extent, data collection will be opportunistic, but it is appropriate to have a well-defined target in mind. We consider the following essential.

Metadata. One means of resource identification is to survey existing documentation for the language, including bibliographic references and locations of web resources. Provenance and proper citation of sources should be included for all data.

For written text. (1) Primary documents in original printed form, e.g. scanned page images or PDF. (2) Transcription. Not only optical character recognition output, but also the output of tools that extract text from PDF, will generally require manual editing.

For spoken text. (1) Audio recordings. Both elicited and spontaneous speech should be included. It is highly desirous to have some connected speech for every language. (2) Slow speech "audio transcriptions." Carefully respeaking a spoken text can be much more efficient than written transcription, and may one day yield to speech recognition methods. (3) Written transcriptions. We do not impose any requirements on the form of transcription, though orthographic transcription is generally much faster to produce than phonetic transcription, and may even be more useful as words are represented by normalized forms.

For both written and spoken text. (1) Translations of primary documents into a reference language (possibly including commentary). (2) Sentence-level segmentation and translation. (3) Word-level segmentation and glossing. (4) Morpheme-level segmentation and glossing.

All documents will be included in primary form, but the percentage of documents with manual annotation, or manually corrected annotation, decreases at increasingly fine-grained levels of annotation. Where manual fine-grained annotation is unavailable, automatic methods for creating it (at a lower quality) are desirable. Defining such methods for a large range of resource-poor languages is an interesting computational challenge.

Secondary resources. Although it is possible to base descriptive analyses exclusively on a text corpus (Himmelmann, 2006, p. 22), the following secondary resources should be secured if they are available: (1) A lexicon with glosses in a reference language. Ideally, everything should be attested in the texts, but as a practical matter, there will be words for which we have only a lexical entry and no instances of use. (2) Paradigms and phonology, for the construction of a morphological analyzer. Ideally, they should be inducible from the texts, but published grammatical information may go beyond what is attested in the text.

2.4 Inadequacy of existing efforts

Our key desideratum is support for automatic processing across a large range of languages. No data collection effort currently exists or is proposed, to our knowledge, that addresses this desideratum. Traditional language archives such as the Audio Archive of Linguistic Fieldwork (UC Berkeley), Documentation of Endangered Languages (Max Planck Institute, Nijmegen), the Endangered Languages Archive (SOAS, University of London), and the Pacific And Regional Archive for Digital Sources in Endangered Cultures (Australia) offer broad coverage of languages, but the majority of their offerings are restricted in availability and do not support machine processing. Conversely, large-scale data collection efforts by the Linguistic Data Consortium and the European Language Resources Association cover less than one percent of the world's languages, with no evident plans for major expansion of coverage. Other efforts concern the definition and aggregation of language resource metadata, including OLAC, IMDI, and

CLARIN (Simons and Bird, 2003; Broeder and Wittenburg, 2006; Váradi et al., 2008), but this is not the same as collecting and disseminating data.

Initiatives to develop standard formats for linguistic annotations are orthogonal to our goals. The success of the project will depend on contributed data from many sources, in many different formats. Converting all data formats to an official standard, such as the RDF-based models being developed by ISO Technical Committee 37 Sub-committee 4 Working Group 2, is simply impractical. These formats have onerous syntactic and semantic requirements that demand substantial further processing together with expert judgment, and threaten to crush the large-scale collaborative data collection effort we envisage, before it even gets off the ground. Instead, we opt for a very lightweight format, sketched in the next section, to minimize the effort of conversion and enable an immediate start. This does not limit the options of community members who desire richer formats, since they are free to invest the effort in enriching the existing data. Such enrichment efforts may gain broad support if they deliver a tangible benefit for cross-language processing.

3 A Simple Storage Model

Here we sketch a simple approach to storage of texts (including transcribed speech), bitexts, interlinear glossed text, and lexicons. We have been deliberately schematic since the goal is just to give grounds for confidence that there exists a general, scalable solution.

For readability, our illustrations will include space-separated sequences of tokens. However, behind the scenes these could be represented as a sequence of pairs of start and end offsets into a primary text or speech signal, or as a sequence of integers that reference an array of strings. Thus, when we write (1a), bear in mind it may be implemented as (1b) or (1c).

(1) a. This is a point of order .
    b. (0,4), (5,7), (8,9), (10,15), (16,18), ...
    c. 9347, 3053, 0038, 3342, 3468, ...

In what follows, we focus on the minimal requirements for storing and disseminating aligned text, not the requirements for efficient in-memory data structures. Moreover, we are agnostic about whether the normalized, tokenized format is stored entire or computed on demand.

We take an aligned text to be composed of a series of aligned sentences, each consisting of a small set of attributes and values, e.g.:

```
ID: europarl/swedish/ep-00-01-17/18
LANGS: swd eng
SENT: det gäller en ordningsfråga
TRANS: this is a point of order
ALIGN: 1-1 2-2 3-3 4-4 4-5 4-6
PROVENANCE: pharaoh-v1.2, ...
REV: 8947 2010-05-02 10:35:06 leobfld12
RIGHTS: Copyright (C) 2010 Uni...; CC-BY
```

The value of `ID` identifies the document and sentence, and any collection to which the document belongs. Individual components of the identifier can be referenced or retrieved. The `LANGS` attribute identifies the source and reference language using ISO 639 codes.[2] The `SENT` attribute contains space-delimited tokens comprising a sentence. Optional attributes `TRANS` and `ALIGN` hold the translation and alignment, if these are available; they are omitted in monolingual text. A provenance attribute records any automatic or manual processes which apply to the record, and a revision attribute contains the version number, timestamp, and username associated with the most recent modification of the record, and a rights attribute contains copyright and license information.

When morphological annotation is available, it is represented by two additional attributes, `LEX` and `AFF`. Here is a monolingual example:

```
ID: example/001
LANGS: eng
SENT: the dogs are barking
LEX: the dog be bark
AFF: - PL PL ING
```

Note that combining all attributes of these two examples—that is, combining word-by-word translation with morphological analysis—yields interlinear glossed text.

A bilingual lexicon is an indispensable resource, whether provided as such, induced from a collection of aligned text, or created by merging contributed and induced lexicons. A bilingual lexicon can be viewed as an inventory of cross-language correspondences between words or groups of words. These correspondences are just aligned text fragments, albeit much smaller than a sentence. Thus, we take a bilingual lexicon to be a kind of text in which each record contains a single lexeme and its translation, represented using the `LEX` and `TRANS` attributes we have already introduced, e.g.:

[2] `http://www.sil.org/iso639-3/`

```
ID: swedishlex/v3.2/0419
LANGS: swd eng
LEX: ordningsfråga
TRANS: point of order
```

In sum, the Universal Corpus is represented as a massive store of records, each representing a single sentence or lexical entry, using a limited set of attributes. The store is indexed for efficient access, and supports access to slices identified by language, content, provenance, rights, and so forth. Many component collections would be "unioned" into this single, large Corpus, with only the record identifiers capturing the distinction between the various data sources.

Special cases of aligned text and wordlists, spanning more than 1,000 languages, are Bible translations and Swadesh wordlists (Resnik et al., 1999; Swadesh, 1955). Here there are obvious use-cases for accessing a particular verse or word across all languages. However, it is not necessary to model n-way language alignments. Instead, such sources are implicitly aligned by virtue of their structure. Extracting all translations of a verse, or all cognates of a Swadesh wordlist item, is an index operation that returns monolingual records, e.g.:

```
ID: swadesh/47          ID: swadesh/47
LANGS: fra              LANGS: eng
LEX: chien              LEX: dog
```

4 Building the Corpus

Data collection on this scale is a daunting prospect, yet it is important to avoid the paralysis of over-planning. We can start immediately by leveraging existing infrastructure, and the voluntary effort of interested members of the language resources community. One possibility is to found a "Language Commons," an open access repository of language resources hosted in the Internet Archive, with a lightweight method for community members to contribute data sets.

A fully processed and indexed version of selected data can be made accessible via a web services interface to a major cloud storage facility, such as Amazon Web Services. A common query interface could be supported via APIs in multiple NLP toolkits such as NLTK and GATE (Bird et al., 2009; Cunningham et al., 2002), and also in generic frameworks such as UIMA and SOAP, leaving developers to work within their preferred environment.

4.1 Motivation for data providers

We hope that potential contributors of data will be motivated to participate primarily by agreement with the goals of the project. Even someone who has specialized in a particular language or language family maintains an interest, we expect, in the universal question—the exploration of Language writ large.

Data providers will find benefit in the availability of volunteers for crowd-sourcing, and tools for (semi-)automated quality control, refinement, and presentation of data. For example, a data holder should be able to contribute recordings and get help in transcribing them, through a combination of volunteer labor and automatic processing.

Documentary linguists and computational linguists have much to gain from collaboration. In return for the data that documentary linguistics can provide, computational linguistics has the potential to revolutionize the tools and practice of language documentation.

We also seek collaboration with communities of language speakers. The corpus provides an economy of scale for the development of literacy materials and tools for interactive language instruction, in support of language preservation and revitalization. For small languages, literacy in the mother tongue is often defended on the grounds that it provides the best route to literacy in the national language (Wagner, 1993, ch. 8). An essential ingredient of any local literacy program is to have a substantial quantity of available texts that represent familiar topics including cultural heritage, folklore, personal narratives, and current events. Transition to literacy in a language of wider communication is aided when transitional materials are available (Waters, 1998, pp. 61ff). Mutual benefits will also flow from the development of tools for low-cost publication and broadcast in the language, with copies of the published or broadcast material licensed to and archived in the corpus.

4.2 Roles

The enterprise requires collaboration of many individuals and groups, in a variety of roles.

Editors. A critical group are people with sufficient engagement to serve as editors for particular language families, who have access to data or are able to negotiate redistribution rights, and oversee the workflow of transcription, translation, and annotation.

CL Research. All manual annotation steps need to be automated. Each step presents a challenging semi-supervised learning and cross-linguistic bootstrapping problem. In addition, the overall measure of success—induction of machine translation systems from limited resources—pushes the state of the art (Kumar et al., 2007). Numerous other CL problems arise: active learning to improve the quality of alignments and bilingual lexicons; automatic language identification for low-density languages; and morphology learning.

Tool builders. We need tools for annotation, format conversion, spidering and language identification, search, archiving, and presentation. Innovative crowd-sourcing solutions are of particular interest, e.g. web-based functionality for transcribing audio and video of oral literature, or setting up a translation service based on aligned texts for a low-density language, and collecting the improved translations suggested by users.

Volunteer annotators. An important reason for keeping the data model as lightweight as possible is to enable contributions from volunteers with little or no linguistic training. Two models are the volunteers who scan documents and correct OCR output in Project Gutenberg, or the undergraduate volunteers who have constructed Greek and Latin treebanks within Project Perseus (Crane, 2010). Bilingual lexicons that have been extracted from aligned text collections might be corrected using crowd-sourcing, leading to improved translation models and improved alignments. We also see the Universal Corpus as an excellent opportunity for undergraduates to participate in research, and for native speakers to participate in the preservation of their language.

Documentary linguists. The collection protocol known as Basic Oral Language Documentation (BOLD) enables documentary linguists to collect 2–3 orders of magnitude more oral discourse than before (Bird, 2010). Linguists can equip local speakers to collect written texts, then to carefully "respeak" and orally translate the texts into a reference language. With suitable tools, incorporating active learning, local speakers could further curate bilingual texts and lexicons. An early need is pilot studies to determine costings for different categories of language.

Data agencies. The LDC and ELRA have a central role to play, given their track record in obtaining, curating, and publishing data with licenses that facilitate language technology development. We need to identify key resources where negotiation with the original data provider, and where payment of all preparation costs plus compensation for lost revenue, leads to new material for the Corpus. This is a new publication model and a new business model, but it can co-exist with the existing models.

Language archives. Language archives have a special role to play as holders of unique materials. They could contribute existing data in its native format, for other participants to process. They could give bilingual texts a distinct status within their collections, to facilitate discovery.

Funding agencies. To be successful, the Human Language Project would require substantial funds, possibly drawing on a constellation of public and private agencies in many countries. However, in the spirit of starting small, and starting now, agencies could require that sponsored projects which collect texts and build lexicons contribute them to the Language Commons. After all, the most effective time to do translation, alignment, and lexicon work is often at the point when primary data is first collected, and this extra work promises direct benefits to the individual project.

4.3 Early tasks

Seed corpus. The central challenge, we believe, is getting critical mass. Data attracts data, and if one can establish a sufficient seed, the effort will snowball. We can make some concrete proposals as to how to collect a seed. Language resources on the web are one source—the Crúbadán project has identified resources for 400 languages, for example (Scannell, 2008); the New Testament of the Bible exists in about 1200 languages and contains of the order of 100k words. We hope that existing efforts that are already well-disposed toward electronic distribution will participate. We particularly mention the Language and Culture Archive of the Summer Institute of Linguistics, and the Rosetta Project. The latter is already distributed through the Internet Archive and contains material for 2500 languages.

Resource discovery. Existing language resources need to be documented, a large un-

dertaking that depends on widely distributed knowledge. Existing published corpora from the LDC, ELRA and dozens of other sources—a total of 85,000 items—are already documented in the combined catalog of the Open Language Archives Community,[3] so there is no need to recreate this information. Other resources can be logged by community members using a public access wiki, with a metadata template to ensure key fields are elicited such as resource owner, license, ISO 639 language code(s), and data type. This information can itself be curated and stored in the form of an OLAC archive, to permit search over the union of the existing and newly documented items. Work along these lines has already been initiated by LDC and ELRA (Cieri et al., 2010).

Resource classification. Editors with knowledge of particular language families will categorize documented resources relative to the needs of the project, using controlled vocabularies. This involves examining a resource, determining the granularity and provenance of the segmentation and alignment, checking its ISO 639 classifications, assigning it to a logarithmic size category, documenting its format and layout, collecting sample files, and assigning a priority score.

Acquisition. Where necessary, permission will be sought to lodge the resource in the repository. Funding may be required to buy the rights to the resource from its owner, as compensation for lost revenue from future data sales. Funding may be required to translate the source into a reference language. The repository's ingestion process is followed, and the resource metadata is updated.

Text collection. Languages for which the available resources are inadequate are identified, and the needs are prioritized, based on linguistic and geographical diversity. Sponsorship is sought for collecting bilingual texts in high priority languages. Workflows are developed for languages based on a variety of factors, such as availability of educated people with native-level proficiency in their mother tongue and good knowledge of a reference language, internet access in the language area, availability of expatriate speakers in a first-world context, and so forth. A classification scheme is required to help predict which workflows will be most successful in a given situation.

[3] http://www.language-archives.org/

Audio protocol. The challenge posed by languages with no written literature should not be underestimated. A promising collection method is Basic Oral Language Documentation, which calls for inexpensive voice recorders and netbooks, project-specific software for transcription and sentence-aligned translation, network bandwidth for upload to the repository, and suitable training and support throughout the process.

Corpus readers. Software developers will inspect the file formats and identify high priority formats based on information about resource priorities and sizes. They will code a corpus reader, an open source reference implementation for converting between corpus formats and the storage model presented in section 3.

4.4 Further challenges

There are many additional difficulties that could be listed, though we expect they can be addressed over time, once a sufficient seed corpus is established. Two particular issues deserve further comment, however.

Licenses. Intellectual property issues surrounding linguistic corpora present a complex and evolving landscape (DiPersio, 2010). For users, it would be ideal for all materials to be available under a single license that permits derivative works, commercial use, and redistribution, such as the Creative Commons Attribution License (CC-BY). There would be no confusion about permissible uses of subsets and aggregates of the collected corpora, and it would be easy to view the Universal Corpus as a single corpus. But to attract as many data contributors as possible, we cannot make such a license a condition of contribution.

Instead, we propose to distinguish between: (1) a digital Archive of contributed corpora that are stored in their original format and made available under a range of licenses, offering preservation and dissemination services to the language resources community at large (i.e. the Language Commons); and (2) the Universal Corpus, which is embodied as programmatic access to an evolving subset of materials from the archive under one of a small set of permissive licenses, licenses whose unions and intersections are understood (e.g. CC-BY and its non-commercial counterpart CC-BY-NC). Apart from being a useful service in its own right, the Archive would provide a staging

ground for the Universal Corpus. Archived corpora having restrictive licenses could be evaluated for their potential as contributions to the Corpus, making it possible to prioritize the work of negotiating more liberal licenses.

There are reasons to distinguish Archive and Corpus even beyond the license issues. The Corpus, but not the Archive, is limited to the formats that support automatic cross-linguistic processing. Conversely, since the primary interface to the Corpus is programmatic, it may include materials that are hosted in many different archives; it only needs to know how to access and deliver them to the user. Incidentally, we consider it an implementation issue whether the Corpus is provided as a web service, a download service with user-side software, user-side software with data delivered on physical media, or a cloud application with user programs executed server-side.

Expenses of conversion and editing. We do not trivialize the work involved in converting documents to the formats of section 3, and in manually correcting the results of noisy automatic processes such as optical character recognition. Indeed, the amount of work involved is one motivation for the lengths to which we have gone to keep the data format simple. For example, we have deliberately avoided specifying any particular tokenization scheme. Variation will arise as a consequence, but we believe that it will be no worse than the variability in input that current machine translation training methods routinely deal with, and will not greatly injure the utility of the Corpus. The utter simplicity of the formats also widens the pool of potential volunteers for doing the manual work that is required. By avoiding linguistically delicate annotation, we can take advantage of motivated but untrained volunteers such as students and members of speaker communities.

5 Conclusion

Nearly twenty years ago, the linguistics community received a wake-up call, when Hale et al. (1992) predicted that 90% of the world's linguistic diversity would be lost or moribund by the year 2100, and warned that linguistics might "go down in history as the only science that presided obliviously over the disappearance of 90 per cent of the very field to which it is dedicated." Today, language documentation is a high priority in mainstream linguistics. However, the field of computational linguistics is yet to participate substantially.

The first half century of research in computational linguistics—from circa 1960 up to the present—has touched on less than 1% of the world's languages. For a field which is justly proud of its empirical methods, it is time to apply those methods to the remaining 99% of languages. We will never have the luxury of richly annotated data for these languages, so we are forced to ask ourselves: can we do more with less?

We believe the answer is "yes," and so we challenge the computational linguistics community to adopt a scalable computational approach to the problem. We need leaner methods for building machine translation systems; new algorithms for cross-linguistic bootstrapping via multiple paths; more effective techniques for leveraging human effort in labeling data; scalable ways to get bilingual text for unwritten languages; and large scale social engineering to make it all happen quickly.

To believe we can build this Universal Corpus is certainly audacious, but not to even try is arguably irresponsible. The initial step parallels earlier efforts to create large machine-readable text collections which began in the 1960s and reverberated through each subsequent decade. Collecting bilingual texts is an orthodox activity, and many alternative conceptions of a Human Language Project would likely include this as an early task.

The undertaking ranks with the largest data-collection efforts in science today. It is not achievable without considerable computational sophistication and the full engagement of the field of computational linguistics. Yet we require no fundamentally new technologies. We can build on our strengths in corpus-based methods, linguistic models, human- and machine-supplied annotations, and learning algorithms. By rising to this, the greatest language challenge of our time, we enable multi-lingual technology development at a new scale, and simultaneously lay the foundations for a new science of empirical universal linguistics.

Acknowledgments

We are grateful to Ed Bice, Doug Oard, Gary Simons, participants of the Language Commons working group meeting in Boston, students in the "Digitizing Languages" seminar (University of Michigan), and anonymous reviewers, for feedback on an earlier version of this paper.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media. http://nltk.org/book.

Steven Bird. 2010. A scalable method for preserving oral literature from small languages. In *Proceedings of the 12th International Conference on Asia-Pacific Digital Libraries*, pages 5–14.

Daan Broeder and Peter Wittenburg. 2006. The IMDI metadata framework, its current application and future direction. *International Journal of Metadata, Semantics and Ontologies*, 1:119–132.

Christopher Cieri, Khalid Choukri, Nicoletta Calzolari, D. Terence Langendoen, Johannes Leveling, Martha Palmer, Nancy Ide, and James Pustejovsky. 2010. A road map for interoperable language resource metadata. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*.

Gregory R. Crane. 2010. Perseus Digital Library: Research in 2008/09. http://www.perseus.tufts.edu/hopper/research/current. Accessed Feb. 2010.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: an architecture for development of robust HLT applications. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175. Association for Computational Linguistics.

Denise DiPersio. 2010. Implications of a permissions culture on the development and distribution of language resources. In *FLaReNet Forum 2010*. Fostering Language Resources Network. http://www.flarenet.eu/.

Hale, M. Krauss, L. Watahomigie, A. Yamamoto, and C. Craig. 1992. Endangered languages. *Language*, 68(1):1–42.

Nikolaus P. Himmelmann. 2006. Language documentation: What is it and what is it good for? In Jost Gippert, Nikolaus Himmelmann, and Ulrike Mosel, editors, *Essentials of Language Documentation*, pages 1–30. Mouton de Gruyter.

Human Genome Project. 2007. The science behind the Human Genome Project. http://www.ornl.gov/sci/techresources/Human_Genome/project/info.shtml. Accessed Dec. 2007.

W. John Hutchins and Harold L. Somers. 1992. *An Introduction to Machine Translation*. Academic Press.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2007. Improving word alignment with bridge languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 42–50, Prague, Czech Republic. Association for Computational Linguistics.

Mike Maxwell and Baden Hughes. 2006. Frontiers in linguistic annotation for lower-density languages. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 29–37, Sydney, Australia, July. Association for Computational Linguistics.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1999. The Bible as a parallel corpus: Annotating the 'book of 2000 tongues'. *Computers and the Humanities*, 33:129–153.

Kevin Scannell. 2008. The Crúbadán Project: Corpus building for under-resourced languages. In *Cahiers du Cental 5: Proceedings of the 3rd Web as Corpus Workshop*.

Gary Simons and Steven Bird. 2003. The Open Language Archives Community: An infrastructure for distributed archiving of language resources. *Literary and Linguistic Computing*, 18:117–128.

Morris Swadesh. 1955. Towards greater accuracy in lexicostatistic dating. *International Journal of American Linguistics*, 21:121–137.

Tamás Váradi, Steven Krauwer, Peter Wittenburg, Martin Wynne, and Kimmo Koskenniemi. 2008. CLARIN: common language resources and technology infrastructure. In *Proceedings of the Sixth International Language Resources and Evaluation Conference*. European Language Resources Association.

Daniel A. Wagner. 1993. *Literacy, Culture, and Development: Becoming Literate in Morocco*. Cambridge University Press.

Glenys Waters. 1998. *Local Literacies: Theory and Practice*. Summer Institute of Linguistics, Dallas.

Douglas H. Whalen and Gary Simons. 2009. Endangered language families. In *Proceedings of the 1st International Conference on Language Documentation and Conservation*. University of Hawaii. http://hdl.handle.net/10125/5017.

Anthony C. Woodbury. 2010. Language documentation. In Peter K. Austin and Julia Sallabank, editors, *The Cambridge Handbook of Endangered Languages*. Cambridge University Press.

Fei Xia and William D. Lewis. 2007. Multilingual structural projection across interlinearized text. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Bilingual Lexicon Generation Using Non-Aligned Signatures

Daphna Shezaf
Institute of Computer Science
Hebrew University of Jerusalem
daphna.shezaf@mail.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Bilingual lexicons are fundamental resources. Modern automated lexicon generation methods usually require parallel corpora, which are not available for most language pairs. Lexicons can be generated using non-parallel corpora or a pivot language, but such lexicons are noisy. We present an algorithm for generating a high quality lexicon from a noisy one, which only requires an independent corpus for each language. Our algorithm introduces *non-aligned signatures (NAS)*, a cross-lingual word context similarity score that avoids the over-constrained and inefficient nature of alignment-based methods. We use NAS to eliminate incorrect translations from the generated lexicon. We evaluate our method by improving the quality of noisy Spanish-Hebrew lexicons generated from two pivot English lexicons. Our algorithm substantially outperforms other lexicon generation methods.

1 Introduction

Bilingual lexicons are useful for both end users and computerized language processing tasks. They provide, for each source language word or phrase, a set of translations in the target language, and thus they are a basic component of dictionaries, which also include syntactic information, sense division, usage examples, semantic fields, usage guidelines, etc.

Traditionally, when bilingual lexicons are not compiled manually, they are extracted from parallel corpora. However, for most language pairs parallel bilingual corpora either do not exist or are at best small and unrepresentative of the general language.

Bilingual lexicons can be generated using non-parallel corpora or pivot language lexicons (see Section 2). However, such lexicons are noisy. In this paper we present a method for generating a high quality lexicon given such a noisy one. Our evaluation focuses on the pivot language case.

Pivot language approaches deal with the scarcity of bilingual data for most language pairs by relying on the availability of bilingual data for each of the languages in question with a third, *pivot*, language. In practice, this third language is often English.

A naive method for pivot-based lexicon generation goes as follows. For each source headword[1], take its translations to the pivot language using the source-to-pivot lexicon, then for each such translation take its translations to the target language using the pivot-to-target lexicon. This method yields highly noisy ('divergent') lexicons, because lexicons are generally intransitive. This intransitivity stems from polysemy in the pivot language that does not exist in the source language. For example, take French-English-Spanish. The English word *spring* is the translation of the French word *printemps*, but only in the season of year sense. Further translating *spring* into Spanish yields both the correct translation *primavera* and an incorrect one, *resorte* (the elastic object).

To cope with the issue of divergence due to lexical intransitivity, we present an algorithm for assessing the correctness of candidate translations. The algorithm is quite simple to understand and to implement and is computationally efficient. In spite of its simplicity, we are not aware of previous work applying it to our problem.

The algorithm utilizes two monolingual corpora, comparable in their domain but otherwise unrelated, in the source and target languages. It does not need a pivot language corpus. The algorithm comprises two stages: signature genera-

[1] In this paper we focus on single word head entries. Multi-word expressions form a major topic in NLP and their handling is deferred to future work.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 98–107,
Uppsala, Sweden, 11-16 July 2010.

tion and signature ranking. The *signature* of word w is the set of words that co-occur with w most strongly. While co-occurrence scores are used to compute signatures, signatures, unlike context vectors, do not contain the score values. For each given source headword we compute its signature and the signatures of all of its candidate translations. We present the *non-aligned signatures (NAS)* similarity score for signature and use it to rank these translations. *NAS* is based on the number of headword signature words that may be translated using the input noisy lexicon into words in the signature of a candidate translation.

We evaluate our algorithm by generating a bilingual lexicon for Hebrew and Spanish using pivot Hebrew-English and English-Spanish lexicons compiled by a professional publishing house. We show that the algorithm outperforms existing algorithms for handling divergence induced by lexical intransitivity.

2 Previous Work

2.1 Parallel Corpora

Parallel corpora are often used to infer word-oriented machine-readable bilingual lexicons. The texts are aligned to each other, at chunk- and/or word-level. Alignment is generally evaluated by consistency (source words should be translated to a small number of target words over the entire corpus) and minimal shifting (in each occurrence, the source should be aligned to a translation nearby). For a review of such methods see (Lopez, 2008). The limited availability of parallel corpora of sufficient size for most language pairs restricts the usefulness of these methods.

2.2 Pivot Language Without Corpora

2.2.1 Inverse Consultation

Tanaka and Umemura (1994) generated a bilingual lexicon using a pivot language. They approached lexical intransitivity divergence using *Inverse Consultation (IC)*. IC examines the intersection of two pivot language sets: the set of pivot translations of a source-language word w, and the set of pivot translations of each target-language word that is a candidate for being a translation to w. IC generally requires that the intersection set contains at least two words, which are synonyms. For example, the intersection of the English translations of French *printemps* and Spanish *resorte* contains only a single word, *spring*. The intersection for a correct translation pair *printemps* and *primavera* may include two synonym words, *spring* and *springtime*. Variations of this method were proposed by (Kaji and Aizono, 1996; Bond et al., 2001; Paik et al., 2004; Ahn and Frampton, 2006).

One weakness of IC is that it relies on pivot language synonyms to identify correct translations. In the above example, if the relatively rare *springtime* had not existed or was missing from the input lexicons, IC would not have been able to discern that *primavera* is a correct translation. This may result in low recall.

2.2.2 Multiple Pivot Languages

Mausam et al. (2009) used many input bilingual lexicons to create bilingual lexicons for new language pairs. They represent the multiple input lexicons in a single undirected graph, with words from all the lexicons as nodes. The input lexicons translation pairs define the edges in the graph. New translation pairs are inferred based on cycles in the graph, that is, the existence of multiple paths between two words in different languages.

In a sense, this is a generalization of the pivot language idea, where multiple pivots are used. In the example above, if both English and German are used as pivots, *printemps* and *primavera* would be accepted as correct because they are linked by both English *spring* and German *Fruehling*, while *printemps* and *resorte* are not linked by any German pivot. This multiple-pivot idea is similar to *Inverse Consultation* in that multiple pivots are required, but using multiple pivot *languages* frees it from the dependency on rich input lexicons that contain a variety of synonyms. This is replaced, however, with the problem of coming up with multiple suitable input lexicons.

2.2.3 Micro-Structure of Dictionary Entries

Dictionaries published by a single publishing house tend to partition the semantic fields of headwords in the same way. Thus the first translation of some English headword in the English-Spanish and in the English-Hebrew dictionaries would correspond to the same sense of the headword, and would therefore constitute translations of each other. The applicability of this method is limited by the availability of machine-readable dictionaries produced by the same publishing house. Not surprisingly, this method has been proposed by lexicographers working in such companies (Sk-

oumalova, 2001).

2.3 Cross-lingual Co-occurrences in Lexicon Construction

Rapp (1999) and Fung (1998) discussed semantic similarity estimation using cross-lingual context vector alignment. Both works rely on a pre-existing large (16-20K entries), correct, one-to-one lexicon between the source and target languages, which is used to align context vectors between languages. The context vector data was extracted from comparable (monolingual but domain-related) corpora. Koehn and Knight (2002) were able to do without the initial large lexicon by limiting themselves to related languages that share a writing system, and using identically-spelled words as context words. Garera et al. (2009) and Pekar et al. (2006) suggested different methods for improving the context vectors data in each language before aligning them. Garera et al. (2009) replaced the traditional window-based co-occurrence counting with dependency-tree based counting, while Pekar et al. (2006) predicted missing co-occurrence values based on similar words in the same language. In the latter work, the one-to-one lexicon assumption was not made: when a context word had multiple equivalents, it was mapped into all of them, with the original probability equally distributed between them.

Pivot Language. Using cross-lingual co-occurrences to improve a lexicon generated using a pivot language was suggested by Tanaka and Iwasaki (1996). Schafer and Yarowsky (2002) created lexicons between English and a target local language (e.g. Gujarati) using a related language (e.g. Hindi) as pivot. An English pivot lexicon was used in conjunction with pivot-target cognates. Cross-lingual co-occurrences were used to remove errors, together with other cues such as edit distance and Inverse Document Frequencies (IDF) scores. It appears that this work assumed a single alignment was possible from English to the target language.

Kaji et al. (2008) used a pivot English lexicon to generate initial Japanese-Chinese and Chinese-Japanese lexicons, then used co-occurrences information, aligned using the initial lexicon, to identify correct translations. Unlike other works, which require alignments of pairs (i.e., two co-occurring words in one language translatable into two co-occurring words in the other), this method relies on alignments of 3-word cliques in each language, every pair of which frequently co-occurring. This is a relatively rare occurrence, which may explain the low recall rates of their results.

3 Algorithm

Our algorithm transforms a noisy lexicon into a high quality one. As explained above, in this paper we focus on noisy lexicons generated using pivot language lexicons. Other methods for obtaining an initial noisy lexicon could be used as well; their evaluation is deferred to future work.

In the setting evaluated in this paper, we first generate an initial noisy lexicon *iLex* possibly containing many translation candidates for each source headword. *iLex* is computed from two pivot-language lexicons, and is the only place in which the algorithm utilizes the pivot language. Afterwards, for each source headword, we compute its signature and the signatures of each of its translation candidates. Signature computation utilizes a monolingual corpus to discover the words that are most strongly related to the word. We now rank the candidates according to the non-aligned signatures (NAS) similarity score, which assesses the similarity between each candidate's signature and that of the headword. For each headword, we select the t translations with the highest NAS scores as correct translations.

3.1 Input Resources

The resources required by our algorithm as evaluated in this paper are: (a) two bilingual lexicons, one from the source to the pivot language and the other from the pivot to the target language. In principle, these two pivot lexicons can be noisy, although in our evaluation we use manually compiled lexicons; (b) two monolingual corpora, one for each of the source and target languages. We have tested the method with corpora of comparable domains, but not covering the same well-defined subjects (the corpora contain news from different countries and over non-identical time periods).

3.2 Initial Lexicon Construction

We create an initial lexicon from the source to the target language using the pivot language: we look up each source language word s in the source-pivot lexicon, and obtain the set P_s of its pivot

translations. We then look up each of the members of P_s in the pivot-target lexicon, and obtain a set T_s of candidate target translations. *iLex* is therefore a mapping from the set of source headwords to the set of candidate target translations. Note that it is possible that not all target lexicon words appear as translation candidates. To create a target to source lexicon, we repeat the process with the directions reversed.

3.3 Signatures

The *signature* of a word w in a language is the set of *N* words most strongly related to w. There are various possible ways to formalize this notion. We use a common and simple one, the words having the highest tendency to co-occur with *w* in a corpus. We count co-occurrences using a sliding fixed-length window of size *k*. We compute, for each pair of words, their Pointwise Mutual Information (PMI), that is:

$$PMI(w_1, w_2) = log\frac{Pr(w_1, w_2)}{Pr(w_1)Pr(w_2)}$$

where $Pr(w_1, w_2)$ is the co-occurrence count, and $Pr(w_i)$ is the total number of appearance of w_i in the corpus (Church and Hanks, 1990). We define the signature $G(w)_{N,k}$ of *w* to be the set of *N* words with the highest PMI with *w*.

Note that a word's signature includes words in the same language. Therefore, two signatures of words in different languages cannot be directly compared; we compare them using a lexicon L as explained below.

Signature is a function of w parameterized by N and k. We discuss the selection of these parameters in section 4.1.5.

3.4 Non-aligned Signatures (NAS) Similarity Scoring

The core strength of our method lies in the way in which we evaluate similarity between words in the source and target languages. For a lexicon *L*, a source word *s* and a target word *t*, $NAS_L(s, t)$ is defined as the number of words in the signature $G(s)_{N,k}$ of *s* that may be translated, using *L*, to words in the signature $G(t)_{N,k}$ of *t*, normalized by dividing it by *N*. Formally,

$$NAS_L(s, t) = \frac{|\{w \in G(s) | L(w) \cap G(t) \neq \emptyset\}|}{N}$$

Where $L(x)$ is the set of candidate translations of x under the lexicon L. Since we use a single lexicon, *iLex*, throughout this work, we usually omit the *L* subscript when referring to NAS.

Language	Sites	Tokens
Hebrew	haartz.co.il, ynet.co.il, nrg.co.il	510M
Spanish	elpais.com, elmundo.com, abc.es	560M

Table 1: Hebrew corpus data.

4 Lexicon Generation Experiments

We tested our algorithm by generating bilingual lexicons for Hebrew and Spanish, using English as a pivot language. We chose a language pair for which basically no parallel corpora exist[2], and that do not share ancestry or writing system in a way that can provide cues for alignment.

We conducted the test twice: once creating a Hebrew-Spanish lexicon, and once creating a Spanish-Hebrew one.

4.1 Experimental Setup

4.1.1 Corpora

The Hebrew and Spanish corpora were extracted from Israeli and Spanish newspaper websites respectively (see table 1 for details). Crawling a small number of sites allowed us to use special-tailored software to extract the textual data from the web pages, thus improving the quality of the extracted texts. Our two corpora are comparable in their domains, news and news commentary.

No kind of preprocessing was used for the Spanish corpus. For Hebrew, closed-class words that are attached to the succeeding word (e.g., 'the', 'and', 'in') were segmented using a simple unsupervised method (Dinur et al., 2009). This method compares the corpus frequencies of the non-prefixed form *x* and the prefixed form *wx*. If *x* is frequent enough, it is assumed to be the correct form, and all the occurrences of *wx* are segmented into two tokens, *w x*. This method was chosen for being simple and effective. However, the segmentation it produces is not perfect. It is context insensitive, segmenting all appearances of a token in the same way, while many *wx* forms are actually ambiguous. Even unambiguous token segmentations may fail when the non-segmented form is very frequent in the domain.

[2]Old testament corpora are for biblical Hebrew, which is very different from modern Hebrew.

Lexicon	# headwords	BF
Eng-Spa	55057	2.4
Spa-Eng	44349	2.9
Eng-Heb	48857	2.5
Heb-Eng	33439	3.7
Spa-Heb	34077	12.6
Heb-Spa	27591	14.8

Table 2: Number of words in lexicons, and branching factors (BF).

Hebrew orthography presents additional difficulties: there are relatively many homographs, and spelling is not quite standardized. These considerations lead us to believe that our choice of language pair is more challenging than, for example, a pair of European languages.

4.1.2 Lexicons

The source of the Hebrew-English lexicon was the Babylon on-line dictionary[3]. For Spanish-English, we used the union of Babylon with the Oxford English-Spanish lexicon. Since the corpus was segmented to words using spaces, lexicon entries containing spaces were discarded.

Lexicon directionality was ignored. All translation pairs extracted for Hebrew-Spanish via English, were also reversed and added to the Spanish-Hebrew lexicon, and vice-versa. Therefore, every *L1-L2* lexicon we mention is identical to the corresponding *L2-L1* lexicon in the set of translation pairs it contains. Our lexicon is thus the 'noisiest' that can be generated using a pivot language and two source-pivot-target lexicons, but it also provides the most complete candidate set possible. Ignoring directionality is also in accordance with the *reversibility principle* of the lexicographic literature (Tomaszczyk, 1998).

Table 2 details the sizes and branching factors (BF) (the average number of translations for headword) of the input lexicons, as well as those of the generated initial noisy lexicon.

4.1.3 Baseline

The performance of our method was compared to three baselines: Inverse Consultation (IC), average cosine distance, and average city block distance. The first is a completely different algorithm, and the last two are a version of our algorithm in which the NAS score is replaced by other scores.

IC (see section 2.2.1) is a corpus-less method. It ranks $t_1, t_2, ...$, the candidate translations of a source word s, by the size of the intersections of the sets of *pivot* translations of t_i and s. Note that IC ranking is a partial order, as the intersection size may be the same for many candidate translations. IC is a baseline for our algorithm as a whole.

Cosine and city block distances are widely used methods for calculating distances of vectors within the same vector space. They are defined here as[4]

$$Cosine(v,u) = 1 - \frac{\sum v_i u_i}{\sqrt{\sum v_i \sum u_i}}$$

$$CityBlock(v,u) = -\sum_i |v_i - u_i|$$

In the case of context vectors, the vector indices, or keys, are words, and their values are co-occurrence based scores. We used the words in our signatures as context vector keys, and PMI scores as values. In this way, the two scores are 'plugged' into our method and serve as baselines for our NAS similarity score.

Since the context vectors are in different languages, we had to translate, or align, the baseline context vectors for the source and target words. Our initial lexicon is a many-to-many relation, so multiple alignments were possible; in fact, the number of possible alignments tends to be very large[5]. We therefore generated *M* random possible alignments, and used the average distance metric across these alignments.

4.1.4 Test Sets and Gold Standard

Following other works (e.g. (Rapp, 1999)), and to simplify the experimental setup, we focused in our experiments on nouns.

A *p-q frequency range* in a corpus is the set of tokens in the places between p and q in the list of corpus tokens, sorted by frequency from high to low. Two types of test sets were used. The first (R1) includes all the singular, correctly segmented (in Hebrew) nouns among the 500 words in the 1001-1500 frequency range. The 1000 highest-frequency tokens were discarded, as a large number of these are utilized as auxiliary syntactic

[3] www.babylon.com.

[4] We modified the standard cosine and city block metrics so that for all measures higher values would be better.

[5] This is another advantage of our NAS score.

	R1		R2	
	Precision	Recall	Precision	Recall
NAS	**82.1%**	**100%**	**56%**	**100%**
Cosine	60.7%	100%	28%	100%
City block	56.3%	100%	32%	100%
IC	55.2%	85.7%	52%	88%

Table 3: Hebrew-Spanish lexicon generation: highest-ranking translation.

	R1		R2	
	Precision	Recall	Precision	Recall
NAS	**87.6%**	**100%**	**80%**	**100%**
Cosine	68%	100%	44%	100%
City block	69.8%	100%	36%	100%
IC	76.4%	100%	48%	92%

Table 4: Spanish-Hebrew Lexicon Generation: highest-ranking translation.

words. This yielded a test set of 112 Hebrew nouns and 169 Spanish nouns. The second (R2), contains 25 words for each of the two languages, obtained by randomly selecting 5 singular correctly segmented nouns from each of the 5 frequency ranges 1-1000 to 4001-5000.

For each of the test words, the correct translations were extracted from a modern professional concise printed Hebrew-Spanish-Hebrew dictionary (Prolog, 2003). This dictionary almost always provides a single Spanish translation for Hebrew headwords. Spanish headwords had 1.98 Hebrew translations on the average. In both cases this is a small number of correct translation comparing to what we might expect with other evaluation methods; therefore this evaluation amounts to a relatively high standard of correctness. Our score comparison experiments (section 5) extend the evaluation beyond this gold standard.

4.1.5 Parameters

The following parameter values were used. The window size for co-occurrence counting, k, was 4. This value was chosen in a small pre-test. Signature size N was 200 (see Section 6.1). The number of alignments M for the baseline scores was 100. The number of translations selected for each headword, t, was set to 1 for ease of testing, but see further notes under results.

4.2 Results

Tables 3 and 4 summarize the results of the Hebrew-Spanish and Spanish-Hebrew lexicon generation respectively, for both the R1 and R2 test sets.

In the three co-occurrence based methods, NAS similarity, cosine distance and and city block distance, the highest ranking translation was selected. Recall is always 100% as a translation from the candidate set is always selected, and all of this set is valid. Precision is computed as the number of test words whose selected translation was one of the translations in the gold standard.

IC translations ranking is a partial order, as usually many translations are scored equally. When *all* translations have the same score, IC is effectively undecided. We calculate recall as the percentage of cases in which there was more than one score rank. A result was counted as precise if *any* of the highest-ranking translations was in the gold-standard, even if other translations were equally ranked, creating a bias in favor of IC.

In both of the Hebrew-Spanish and the Spanish-Hebrew cases, our method significantly outperformed all baselines in generating a precise lexicon on the highest-ranking translations.

All methods performed better in *R1* than in *R2*, which included also lower-frequency words, and this was more noticeable with the corpus-based methods (Hebrew-Spanish) than with IC. This suggests, not surprisingly, that the performance of corpus-based methods is related to the amount of information in the corpus.

That the results for the Spanish-Hebrew lexicon are higher may arise from the difference in the gold standard. As mentioned, Hebrew words only had one "correct" Spanish translation, while Spanish had 1.98 correct translations on the average. If we had used a more comprehensive resource to test against, the precision of the method would be higher than shown here.

In translation pairs generation, the results beyond the top-ranking pair are also of importance. Tables 5 and 6 present the accuracy of the first three translation suggestions, for the three co-occurrence based scores, calculated for the R1 test set. IC results are not included, as they are incomparable to those of the other methods: IC tends to score many candidate translations identically, and in practice, the three highest-scoring sets of translation candidates contained on average 77% of all

	1st	2nd	3rd	total
NAS	**82.1%**	6.3%	1.8%	**90.2%**
Cosine	60.7%	**9.8%**	2.7%	73.2%
City block	56.3%	4.5%	**10.7%**	71.4%

Table 5: Hebrew-Spanish lexicon generation: accuracy of 3 best translations for the R1 condition. The table shows how many of the 2nd and 3rd translations are correct. Note that NAS is always a better solution, even though its numbers for 2nd and 3rd are smaller, because its accumulative percentage, shown in the last column, is higher.

	1st	2nd	3rd	total
NAS	**87.6%**	**77.5%**	**16%**	**163.9%**
Cosine	68%	66.3%	10.1%	144.4%
City block	69.8%	64.5%	**7.7%**	142%

Table 6: Spanish-Hebrew lexicon generation: accuracy of 3 best translations for the R1 condition. The total exceeds 100% because Spanish words had more than one correct translation. See also the caption of Table 5.

the candidates, thus necessarily yielding mostly incorrect translations. Recall was omitted from the tables as it is always 100%.

For all methods, many of the correct translations that do not rank first, rank as second or third. For both languages, NAS ranks highest for total accuracy of the three translations, with considerable advantage.

5 Score Comparison Experiments

Lexicon generation, as defined in our experiment, is a relatively high standard for cross-linguistic semantic distance evaluation. This is especially correct since our gold standard gives only a small set of translations. The set of possible translations in *iLex* tends to include, besides the "correct" translation of the gold standard, other translations that are suitable in certain contexts or are semantically related. For example, for one Hebrew word, *kvuza*, the gold standard translation was *grupo* (group), while our method chose *equipo* (team), which was at least as plausible given the amount of sports news in the corpus.

	Heb-Spa		Spa-Heb	
	SCE1	SCE2	SCE1	SCE2
NAS	**93.8%**	**76.2%**	**94.1%**	**83.7%**
Cosine	74.1%	57.1%	70.7%	63.2%
City block	74.1%	68.3%	78,1%	75.2%

Table 7: Precision of score comparison experiments. The percentage of cases in which each of the scoring methods was able to successfully distinguish the *correct* (SCE1) or *possible correct* (SCE2) translation from the *random* translation.

Thus to better compare the capability of NAS to distinguish correct and incorrect translations with that of other scores, we performed two more experiments. In the first score comparison experiment (SCE1), we used the two R1 test sets, Hebrew and Spanish, from the lexicon generation test (section 4.1.4). For each word in the test set, we used our method to select between one of two translations: a *correct translation*, from the gold standard, and a *random* translation, chosen randomly among all the nouns similar in frequency to the correct translation.

The second score comparison experiment (SCE2) was designed to test the score with a more extensive test set. For each of the two languages, we randomly selected 1000 nouns, and used our method to select between a *possibly correct* translation, chosen randomly among the translations suggested in *iLex*, and a *random* translation, chosen randomly among nouns similar in frequency to the *possibly correct* translation. This test, while using a more extensive test set, is less accurate because it is not guaranteed that any of the input translations is correct.

In both SCE1 and SCE2, cosine and city block distance were used as baselines. Inverse Consultation is irrelevant here because it can only score translation pairs that appear in *iLex*.

Table 7 presents the results of the two score comparison experiments, each of them for each of the translation directions. Recall is by definition 100% and is omitted.

Again, NAS performs better than the baselines in all cases. With all scores, precision values in SCE1 are higher than in the lexicon generation experiment. This is consistent with the expectation that selection between a correct and a random, probably incorrect, translation is easier than selecting among the translations in *iLex*. The precision in SCE2 is lower than that in SCE1. This may be a result of both translations in SCE2 being

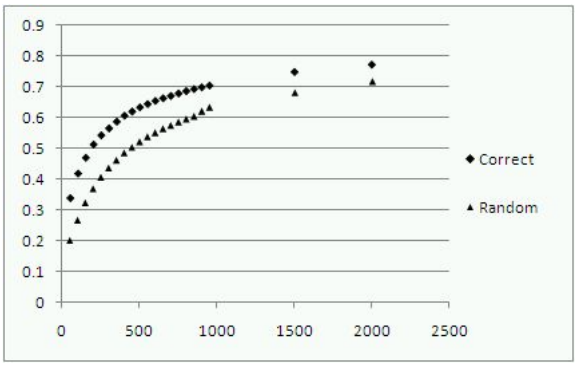


Figure 1: NAS values (not algorithm precision) for various N sizes. NAS is not sensitive to the value of N (see text).

in some cases incorrect. Yet this may also reflect a weakness of all three scores with lower-frequency words, which are represented in the 1000-word samples but not in the ones used in SCE1.

6 NAS Score Properties

6.1 Signature Size

NAS values are in the range $[0, 1]$. The values depend on *N*, the size of the signature used. With an extremely small *N*, NAS values would usually be 0, and would tend to be noisy, due to accidental inclusion of high-frequency or highly ambiguous words in the signature. As *N* approaches the size of the lexicon used for alignment, NAS values approach 1 for all word pairs.

This suggests that choosing a suitable value of *N* is critical for effectively using NAS. Yet an empirical test has shown that NAS may be useful for a wide range of *N* values: we computed NAS values for the *correct* and *random* translations used in the Hebrew-Spanish SCE1 experiment (section 5), using *N* values between 50 and 2000.

Figure 1 shows the average score values (note that these are not precision values) for the correct and random translations across that *N* range. The scores for the correct translations are consistently higher than those of the random translations, even while there is a discernible decline in the difference between them. In fact, the precision of the selection between the correct and random translation is persistent throughout the range. This suggests that while extreme *N* values should be avoided, the selection of *N* is not a major issue.

6.2 Dependency on Alignment Lexicon

NAS_L values depend on L, the lexicon in use. Clearly again, in the extremes, an almost empty lexicon or a lexicon containing every possible pair of words (a Cartesian product), this score would not be useful. In the first case, it would yield 0 for every pair, and in the second, 1. However as our experiments show, it performed well with real-world examples of a noisy lexicon, with branching factors of 12.6 and 14.8 (see table 2).

6.3 Lemmatization

Lemmatization is the process of extracting the lemmas of words in the corpus. Our experiments show that good results can be achieved without lemmatization, at least for nouns in the pair of languages tested (aside from the simple prefix segmentation we used for Hebrew, see section 4.1.1). For other language pairs lemmatization may be needed. In general, correct lemmatization should improve results, since the signatures would consist of more meaningful information. If automatic lemmatization introduces noise, it may reduce the results' quality.

6.4 Alternative Models for Relatedness

Cosine and city block, as well as other related distance metrics, rely on *context vectors*. The context vector of a word *w* collects words and maps them to some score of their "relatedness" to *w*; in this case, we used PMI. NAS, in contrast, relies on the signature, the set of *N* words most related to *w*. That is, it requires a Boolean relatedness indication, rather than a numeric relatedness score. We used PMI to generate this Boolean indication, and naturally, other similar measures could be used as well. More significantly, it may be possible to use it with corpus-less sources of "relatedness", such as WordNet or search result snippets.

7 Conclusion

We presented a method to create a high quality bilingual lexicon given a noisy one. We focused on the case in which the noisy lexicon is created using two pivot language lexicons. Our algorithm uses two unrelated monolingual corpora. At the heart of our method is the non-aligned signatures (NAS) context similarity score, used for removing incorrect translations using cross-lingual co-occurrences.

Words in one language tend to have multiple translations in another. The common method for context similarity scoring utilizes some algebraic distance between context vectors, and requires a single alignment of context vectors in one language into the other. Finding a single correct alignment is unrealistic even when a perfectly correct lexicon is available. For example, alignment forces us to choose one correct translation for each context word, while in practice a few possible terms may be used interchangeably in the other language. In our task, moreover, the lexicon used for alignment was automatically generated from pivot language lexicons and was expected to contain errors.

NAS does not depend on finding a single correct alignment. While it measures how well the sets of words that tend to co-occur with these two words align to each other, its strength may lie in bypassing the question of *which* word in one language should be aligned to a certain context word in the other language. Therefore, unlike other scoring methods, it is not effected by incorrect alignments.

We have shown that NAS outperforms the more traditional distance metrics, which we adapted to the many-to-many scenario by amortizing across multiple alignments. Our results confirm that alignment is problematic in using co-occurrence methods across languages, at least in our settings. NAS constitutes a way to avoid this problem.

While the purpose of this work was to discern correct translations from incorrect one, it is worth noting that our method actually ranks translation correctness. This is a stronger property, which may render it useful in a wider range of scenarios.

In fact, NAS can be viewed as a general measure for word similarity between languages. It would be interesting to further investigate this observation with other sources of lexicons (e.g., obtained from parallel or comparable corpora) and for other tasks, such as cross-lingual word sense disambiguation and information retrieval.

References

Kisuh Ahn and Matthew Frampton. 2006. Automatic generation of translation dictionaries using intermediary languages. In *EACL 2006 Workshop on Cross-Language Knowledge Induction*.

Francis Bond, Ruhaida Binti Sulong, Takefumi Yamazaki, and Kentaro Ogura. 2001. Design and construction of a machine-tractable japanese-malay dictionary. In *MT Summit VIII: Machine Translation in the Information Age, Proceedings*, pages 53–58.

Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16:22–29.

Elad Dinur, Dmitry Davidov, and Ari Rappoport. 2009. Unsupervised concept discovery in hebrew using simple unsupervised word prefix segmentation for hebrew and arabic. In *EACL 2009 Workshop on Computational Approaches to Semitic Languages*.

Pascale Fung. 1998. A statistical view on bilingual lexicon extraction:from parallel corpora to non-parallel corpora. In *The Third Conference of the Association for Machine Translation in the Americas*.

Nikesh Garera, Chris Callison-Burch, and David Yarowsky. 2009. Improving translation lexicon induction from monolingual corpora via dependency contexts and part-of-speech equivalences. In *CoNLL*.

Hiroyuki Kaji and Toshiko Aizono. 1996. Extracting word correspondences from bilingual corpora based on word co-occurrence information. In *COLING*.

Hiroyuki Kaji, Shin'ichi Tamamura, and Dashtseren Erdenebat. 2008. Automatic construction of a japanese-chinese dictionary via english. In *LREC*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*.

Adam Lopez. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49.

Mausam, Stephen Soderland, Oren Etzioni, Daniel S. Weld, Michael Skinner, and Jeff Bilmes. 2009. Compiling a massive, multilingual dictionary via probabilistic inference. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing*.

Kyonghee Paik, Satoshi Shirai, and Hiromi Nakaiwa. 2004. Automatic construction of a transfer dictionary considering directionality. In *COLING, Multilingual Linguistic Resources Workshop*.

Viktor Pekar, Ruslan Mitkov, Dimitar Blagoev, and Andrea Mulloni. 2006. Finding translations for low-frequency words in comparable corpora. *Machine Translation*, 20:247 – 266.

Prolog. 2003. *Practical Bilingual Dictionary: Spanish-Hebew/Hebrew-Spanish*. Israel.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *ACL*.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *CoNLL*.

Hana Skoumalova. 2001. Bridge dictionaries as bridges between languages. *International Journal of Corpus Linguistics*, 6:95–105.

Kumiko Tanaka and Hideya Iwasaki. 1996. Extraction of lexical translations from non-aligned corpora. In *Conference on Computational linguistics*.

Kumiko Tanaka and Kyoji Umemura. 1994. Construction of a bilingual dictionary intermediated by a third language. In *Conference on Computational Linguistics*.

Jerzy Tomaszczyk. 1998. The bilingual dictionary under review. In *ZuriLEX'86*.

Automatic Evaluation Method for Machine Translation using Noun-Phrase Chunking

Hiroshi Echizen-ya
Hokkai-Gakuen University
S 26-Jo, W 11-chome, Chuo-ku,
Sapporo, 064-0926 Japan
echi@eli.hokkai-s-u.ac.jp

Kenji Araki
Hokkaido University
N 14-Jo, W 9-Chome, Kita-ku,
Sapporo, 060-0814 Japan
araki@media.eng.hokudai.ac.jp

Abstract

As described in this paper, we propose a new automatic evaluation method for machine translation using noun-phrase chunking. Our method correctly determines the matching words between two sentences using corresponding noun phrases. Moreover, our method determines the similarity between two sentences in terms of the noun-phrase order of appearance. Evaluation experiments were conducted to calculate the correlation among human judgments, along with the scores produced using automatic evaluation methods for MT outputs obtained from the 12 machine translation systems in NTCIR-7. Experimental results show that our method obtained the highest correlations among the methods in both sentence-level adequacy and fluency.

1 Introduction

High-quality automatic evaluation has become increasingly important as various machine translation systems have developed. The scores of some automatic evaluation methods can obtain high correlation with human judgment in document-level automatic evaluation(Coughlin, 2007). However, sentence-level automatic evaluation is insufficient. A great gap exists between language processing of automatic evaluation and the processing by humans. Therefore, in recent years, various automatic evaluation methods particularly addressing sentence-level automatic evaluations have been proposed. Methods based on word strings (*e.g.*, BLEU(Papineni et al., 2002), NIST(NIST, 2002), METEOR(Banerjee and Lavie., 2005), ROUGE-L(Lin and Och, 2004), and IMPACT(Echizen-ya and Araki, 2007)) calculate matching scores using only common words between MT outputs and references from bilingual humans. However, these methods cannot determine the correct word correspondences sufficiently because they fail to focus solely on phrase correspondences. Moreover, various methods using syntactic analytical tools(Pozar and Charniak, 2006; Mutton et al., 2007; Mehay and Brew, 2007) are proposed to address the sentence structure. Nevertheless, those methods depend strongly on the quality of the syntactic analytical tools.

As described herein, for use with MT systems, we propose a new automatic evaluation method using noun-phrase chunking to obtain higher sentence-level correlations. Using noun phrases produced by chunking, our method yields the correct word correspondences and determines the similarity between two sentences in terms of the noun phrase order of appearance. Evaluation experiments using MT outputs obtained by 12 machine translation systems in NTCIR-7(Fujii et al., 2008) demonstrate that the scores obtained using our system yield the highest correlation with the human judgments among the automatic evaluation methods in both sentence-level adequacy and fluency. Moreover, the differences between correlation coefficients obtained using our method and other methods are statistically significant at the 5% or lower significance level for adequacy. Results confirmed that our method using noun-phrase chunking is effective for automatic evaluation for machine translation.

2 Automatic Evaluation Method using Noun-Phrase Chunking

The system based on our method has four processes. First, the system determines the corre-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 108–117,
Uppsala, Sweden, 11-16 July 2010.

spondences of noun phrases between MT outputs and references using chunking. Secondly, the system calculates word-level scores based on the correct matched words using the determined correspondences of noun phrases. Next, the system calculates phrase-level scores based on the noun-phrase order of appearance. The system calculates the final scores combining word-level scores and phrase-level scores.

2.1 Correspondence of Noun Phrases by Chunking

The system obtains the noun phrases from each sentence by chunking. It then determines corresponding noun phrases between MT outputs and references calculating the similarity for two noun phrases by the PER score(Su et al., 1992). In that case, PER scores of two kinds are calculated. One is the ratio of the number of match words between an MT output and reference for the number of all words of the MT output. The other is the ratio of the number of match words between the MT output and reference for the number of all words of the reference. The similarity is obtained as an F-measure between two PER scores. The high score represents that the similarity between two noun phrases is high. Figure 1 presents an example of the determination of the corresponding noun phrases.

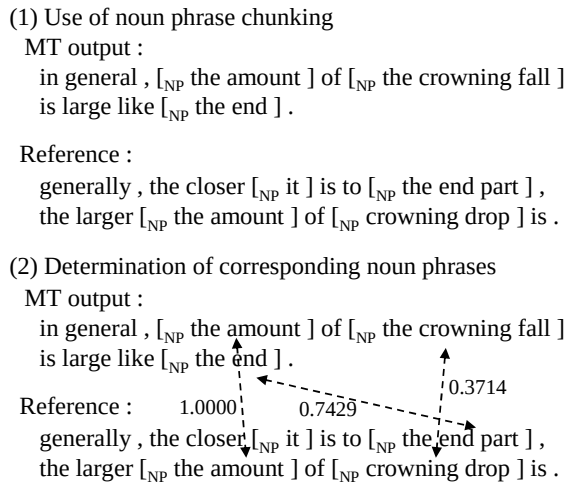


Figure 1: Example of determination of corresponding noun phrases.

In Fig. 1, "the amount", "the crowning fall" and "the end" are obtained as noun phrases in MT output by chunking, and "it", "the end part", "the amount" and "crowning drop" are obtained in the reference by chunking. Next, the system determines the corresponding noun phrases from these noun phrases between the MT output and reference. The score between "the end" and "the end part" is the highest among the scores between "the end" in the MT output and "it", "the end part", "the amount", and "crowning drop" in the reference. Moreover, the score between "the end part" and "the end" is the highest among the scores between "the end part" in reference and "the amount", "the crowning fall", "the end" in the MT output. Consequently, "the end" and "the end part" are selected as noun phrases with the highest mutual scores: "the end" and "the end part" are determined as one corresponding noun phrase. In Fig. 1, "the amount" in the MT output and "the amount" in reference, and "the crowning fall" in the MT output and "crowning drop" in the reference also are determined as the respective corresponding noun phrases. The noun phrase for which the score between it and other noun phrases is 0.0 (*e.g.*, "it" in reference) has no corresponding noun phrase. The use of the noun phrases is effective because the frequency of the noun phrases is higher than those of other phrases. The verb phrases are not used for this study, but they can also be generated by chunking. It is difficult to determine the corresponding verb phrases correctly because the words in each verb phrase are often fewer than the noun phrases.

2.2 Word-level Score

The system calculates the word-level scores between MT output and reference using the corresponding noun phrases. First, the system determines the common words based on Longest Common Subsequence (LCS). The system selects only one LCS route when several LCS routes exist. In such cases, the system calculates the Route Score (RS) using the following Eqs. (1) and (2):

$$RS = \sum_{c \in LCS} \left(\sum_{w \in c} weight(w) \right)^{\beta} \qquad (1)$$

$$weight(w) = \begin{cases} 2 & \text{words in corresponding noun phrase} \\ 1 & \text{words in non-corresponding noun phrase} \end{cases} \quad (2)$$

In Eq. (1), β is a parameter for length weighting of common parts; it is greater than 1.0. Figure 2 portrays an example of determination of the common parts. In the first process of Fig. 2, LCS is 7. In this example, several LCS routes exist. The system selects the LCS route which has ",", "the amount of", "crowning", "is", and "." as the common parts. The common part is the part for which the common words appear continuously. In contrast, IMPACT selects a different LCS route that includes ", the", "amount of", "crowning", "is", and "." as the common parts. In IMPACT, using no analytical knowledge, the LCS route is determined using the information of the number of words in the common parts and the position of the common parts. The RS for LCS route selected using our method is 32 $(= 1^{2.0} + (2 + 2 + 1)^{2.0} + 2^{2.0} + 1^{2.0} + 1^{2.0})$ when β is 2.0. The RS for LCS route selected by IMPACT is 19 $(= (1 + 1)^{2.0} + (2 + 1)^{2.0} + 2^{2.0} + 1^{2.0} + 1^{2.0})$. In the LCS route selected by IMPACT, the weight of "the" in the common part ", the" is 1 because "the" in the reference is not included in the corresponding noun phrase. In the LCS route selected using our method, the weight of "the" in "the amount of" is 2 because "the" in MT output and "the" in the reference are included in the corresponding noun phrase "NP1". Therefore, the system based on our method can select the correct LCS route.

Moreover, the word-level score is calculated using the common parts in the selected LCS route as the following Eqs. (3), (4), and (5).

$$R_{wd} = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in LCS} length(c)^\beta \right)}{m^\beta} \right)^{\frac{1}{\beta}} \quad (3)$$

$$P_{wd} = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in LCS} length(c)^\beta \right)}{n^\beta} \right)^{\frac{1}{\beta}} \quad (4)$$

(1) First process for determination of common parts :
LCS = 7
Our method
MT output :
in general **,** [$_{NP1}$ **the amount] of** [$_{NP2}$ the **crowning** fall] **is** large like [$_{NP3}$ the end] **.**
Reference : $1^{2.0}$ $(2+2+1)^{2.0}$ $1^{2.0}$ $1^{2.0}$ $2^{2.0}$
generally **,** the closer [$_{NP}$ it] is to [$_{NP3}$ the end part] , the larger [$_{NP1}$ **the amount] of** [$_{NP2}$ **crowning** drop] **is .**

IMPACT
MT output :
in general **,** [$_{NP1}$ **the amount] of** [$_{NP2}$ the **crowning** fall] **is** large like [$_{NP3}$ the end] **.**
Reference : $(1+1)^{2.0}$ $(2+1)^{2.0}$ $2^{2.0}$ $1^{2.0}$ $1^{2.0}$
generally **, the** closer [$_{NP}$ it] is to [$_{NP3}$ the end part] , the larger [$_{NP1}$ the **amount] of** [$_{NP2}$ **crowning** drop] **is .**

(2) Second process for determination of common parts :
LCS=3
Our method
MT output :
in general , [$_{NP1}$ *the amount*] *of* [$_{NP2}$ **the** *crowning* fall] *is* large like [$_{NP3}$ **the end**] .
Reference :
generally , **the** closer [$_{NP}$ it] is to [$_{NP3}$ **the end** part] , the larger [$_{NP1}$ *the amount*] *of* [$_{NP2}$ *crowning* drop] *is* .

Figure 2: Example of common-part determination.

$$score_{wd} = \frac{(1 + \gamma^2) R_{wd} P_{wd}}{R_{wd} + \gamma^2 P_{wd}} \quad (5)$$

Equation (3) represents recall and Eq. (4) represents precision. Therein, m signifies the word number of the reference in Eq. (3), and n stands for the word number of the MT output in Eq. (4). Here, RN denotes the repetition number of the determination process of the LCS route, and i, which has initial value 0, is the counter for RN. In Eqs. (3) and (4), α is a parameter for the repetition process of the determination of LCS route, and is less than 1.0. Therefore, R_{wd} and P_{wd} becomes small as the appearance order of the common parts between MT output and reference is different. Moreover, $length(c)$ represents the number of words in each common part; β is a parameter related to the length weight of common parts, as in Eq. (1). In this case, the weight of each common word in the common part is 1. The system calculates $score_{wd}$ as the word-level score in Eq. (5). In Eq. (5), γ is determined as P_{wd}/R_{wd}. The $score_{wd}$ is between 0.0 and 1.0.

In the first process of Fig. 2, $\alpha^i \sum_{c \in LCS} length(c)^\beta$ is 13.0 ($=0.5^0 \times (1^{2.0} + 3^{2.0} + 1^{2.0} + 1^{2.0} + 1^{2.0})$) when α and β are 0.5 and 2.0, respectively. In this case, the counter i is 0. Moreover, in the second process of Fig. 2, $\alpha^i \sum_{c \in LCS} length(c)^\beta$ is 2.5 ($=0.5^1 \times (1^{2.0} + 2^{2.0})$) using two common parts "the" and "the end", except the common parts determined using the first process. In Fig. 2, RN is 1 because the system finishes calculating $\alpha^i \sum_{c \in LCS} length(c)^\beta$ when counter i became 1: this means that all common parts were processed until the second process. As a result, R_{wd} is 0.1969 ($=\sqrt{(13.0 + 2.5)/20^{2.0}} = \sqrt{0.0388}$), and P_{wd} is 0.2625 ($=\sqrt{(13.0 + 2.5)/15^{2.0}} = \sqrt{0.0689}$). Consequently, $score_{wd}$ is 0.2164 ($=\frac{(1+1.3332^2) \times 0.1969 \times 0.2625}{0.1969 + 1.3332^2 \times 0.2625}$). In this case, γ becomes 1.3332 ($=\frac{0.2625}{0.1969}$). The system can determine the matching words correctly using the corresponding noun phrases between the MT output and the reference.

The system calculates $score_{wd\text{-}multi}$ using $R_{wd\text{-}multi}$ and $P_{wd\text{-}multi}$ which are, respectively, maximum R_{wd} and P_{wd} when multiple references are used as the following Eqs. (6), (7) and (8). In Eq. (8), γ is determined as $P_{wd\text{-}multi}/R_{wd\text{-}multi}$. The $score_{wd\text{-}multi}$ is between 0.0 and 1.0.

$$R_{wd\text{-}multi} = \max_{j=1}^{u} \left(\left(\frac{\left(\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in LCS} length(c)^\beta \right) \right)_j}{m_j^\beta} \right)^{\frac{1}{\beta}} \right) \quad (6)$$

$$P_{wd\text{-}multi} = \max_{j=1}^{u} \left(\left(\frac{\left(\sum_{i=0}^{RN} \left(\alpha^i \sum_{c \in LCS} length(c)^\beta \right) \right)_j}{n_j^\beta} \right)^{\frac{1}{\beta}} \right) \quad (7)$$

$$score_{wd\text{-}multi} = \frac{(1 + \gamma^2 R_{wd\text{-}multi}) P_{wd\text{-}multi}}{R_{wd\text{-}multi} + \gamma^2 P_{wd\text{-}multi}} \quad (8)$$

2.3 Phrase-level Score

The system calculates the phrase-level score using the noun phrases obtained by chunking. First, the system extracts only noun phrases from sentences. Then it generalizes each noun phrase as each word. Figure 3 presents examples of generalization by noun phrases.

(1) Corresponding noun phrases
MT output :
in general , [$_{NP1}$ the amount] of [$_{NP2}$ the crowning fall] is large like [$_{NP3}$ the end] .
Reference :
generally , the closer [$_{NP}$ it] is to [$_{NP3}$ the end part] , the larger [$_{NP1}$ the amount] of [$_{NP2}$ crowning drop] is .
(2) Generalization by noun phrases
MT output :
NP1 NP2 NP3
Reference :
NP NP3 NP1 NP2

Figure 3: Example of generalization by noun phrases.

Figure 3 presents three corresponding noun phrases between the MT output and the reference. The noun phrase "it", which has no corresponding noun phrase, is expressed as "NP" in the reference. Consequently, the MT output is generalized as "NP1 NP2 NP3"; the reference is generalized as "NP NP3 NP1 NP2". Subsequently, the system obtains the phrase-level score between the generalized MT output and reference as the following Eqs. (9), (10), and (11).

$$R_{np} = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{cnpp \in LCS} length(cnpp)^\beta \right)}{\left(m_{cnp} \times \sqrt{m_{no\text{-}cnp}} \right)^\beta} \right)^{\frac{1}{\beta}} \quad (9)$$

$$P_{np} = \left(\frac{\sum_{i=0}^{RN} \left(\alpha^i \sum_{cnpp \in LCS} length(cnpp)^\beta \right)}{\left(n_{cnp} \times \sqrt{n_{no\text{-}cnp}} \right)^\beta} \right)^{\frac{1}{\beta}} \quad (10)$$

Table 1: Machine translation system types.

	System No. 1	System No. 2	System No. 3	System No. 4	System No. 5	System No. 6
Type	SMT	SMT	RBMT	SMT	SMT	SMT
	System No. 7	System No. 8	System No. 9	System No. 10	System No. 11	System No. 12
Type	SMT	SMT	EBMT	SMT	SMT	RBMT

$$score_{np} = \frac{(1+\gamma^2)R_{np}P_{np}}{R_{np}+\gamma^2 P_{np}} \quad (11)$$

In Eqs. (9) and (10), *cnpp* denotes the common noun phrase parts; m_{cnp} and n_{cnp} respectively signify the quantities of common noun phrases in the reference and MT output. Moreover, $m_{no\text{-}cnp}$ and $n_{no\text{-}cnp}$ are the quantities of noun phrases except the common noun phrases in the reference and MT output. The values of $m_{no\text{-}cnp}$ and $n_{no\text{-}cnp}$ are processed as 1 when no non-corresponding noun phrases exist. The square root used for $m_{no\text{-}cnp}$ and $n_{no\text{-}cnp}$ is to decrease the weight of the non-corresponding noun phrases. In Eq. (11), γ is determined as P_{np}/R_{np}. In Fig. 3, R_{np} and P_{np} are 0.7071 ($=\sqrt{\frac{1\times 2^{2.0}+0.5\times 1^{2.0}}{(3\times 1)^{2.0}}}$) when α is 0.5 and β is 2.0. Therefore, $score_{np}$ is 0.7071.

The system obtains $score_{np\text{-}multi}$ calculating the average of $score_{np}$ when multiple references are used as the following Eq. (12).

$$score_{np\text{-}multi} = \frac{\sum_{j=0}^{u}\left(score_{np}\right)_j}{u} \quad (12)$$

2.4 Final Score

The system calculates the final score by combining the word-level score and the phrase-level score as shown in the following Eq. (13).

$$score = \frac{score_{wd} + \delta \times score_{np}}{1+\delta} \quad (13)$$

Therein, δ represents a parameter for the weight of $score_{np}$: it is between 0.0 and 1.0. The ratio of $score_{wd}$ to $score_{np}$ is 1:1 when δ is 1.0. Moreover, $score_{wd\text{-}multi}$ and $score_{np\text{-}multi}$ are used for Eq. (13) in multiple references. In Figs. 2 and 3, the final score between the MT output and the reference is 0.4185 ($=\frac{0.2164+0.7\times 0.7071}{1+0.7}$) when δ is 0.7. The system can realize high-quality automatic evaluation using both word-level information and phrase-level information.

3 Experiments

3.1 Experimental Procedure

We calculated the correlation between the scores obtained using our method and scores produced by human judgment. The system based on our method obtained the evaluation scores for 1,200 English output sentences related to the patent sentences. These English output sentences are sentences that 12 machine translation systems in NTCIR-7 translated from 100 Japanese sentences. Moreover, the number of references to each English sentence in 100 English sentences is four. These references were obtained from four bilingual humans. Table 1 presents types of the 12 machine translation systems.

Moreover, three human judges evaluated 1,200 English output sentences from the perspective of adequacy and fluency on a scale of 1–5. We used the median value in the evaluation results of three human judges as the final scores of 1–5. We calculated Pearson's correlation efficient and Spearman's rank correlation efficient between the scores obtained using our method and the scores by human judgments in terms of sentence-level adequacy and fluency.

Additionally, we calculated the correlations between the scores using seven other methods and the scores by human judgments to compare our method with other automatic evaluation methods. The other seven methods were IMPACT, ROUGE-L, BLEU[1], NIST, NMG-WN(Ehara, 2007; Echizen-ya et al., 2009), METEOR[2], and WER(Leusch et al., 2003). Using our method, 0.1 was used as the value of the parameter α in Eqs. (3)-(10) and 1.1 was used as the value of the parameter β in Eqs. (1)–(10). Moreover, 0.3 was used as the value of the parameter δ in Eq. (13). These val-

[1]BLEU was improved to perform sentence-level evaluation: the maximum N value between MT output and reference is used(Echizen-ya et al., 2009).

[2]The matching modules of METEOR are the exact and stemmed matching module, and a WordNet-based synonym-matching module.

Table 2: Pearson's correlation coefficient for sentence-level adequacy.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7
Our method	**0.7862**	**0.4989**	0.5970	**0.5713**	**0.6581**	**0.6779**	**0.7682**
IMPACT	0.7639	0.4487	0.5980	0.5371	0.6371	0.6255	0.7249
ROUGE-L	0.7597	0.4264	**0.6111**	0.5229	0.6183	0.5927	0.7079
BLEU	0.6473	0.2463	0.4230	0.4336	0.3727	0.4124	0.5340
NIST	0.5135	0.2756	0.4142	0.3086	0.2553	0.2300	0.3628
NMG-WN	0.7010	0.3432	0.6067	0.4719	0.5441	0.5885	0.5906
METEOR	0.4509	0.0892	0.3907	0.2781	0.3120	0.2744	0.3937
WER	0.7464	0.4114	0.5519	0.5185	0.5461	0.5970	0.6902
Our method II	0.7870	0.5066	0.5967	0.5191	0.6529	0.6635	0.7698
BLEU with our method	0.7244	0.3935	0.5148	0.5231	0.4882	0.5554	0.6459
	No. 8	No. 9	No. 10	No. 11	No. 12	Avg.	All
Our method	**0.7664**	**0.7208**	**0.6355**	**0.7781**	0.5707	**0.6691**	**0.6846**
IMPACT	0.7007	0.7125	0.5981	0.7621	0.5345	0.6369	0.6574
ROUGE-L	0.6834	0.7042	0.5691	0.7480	0.5293	0.6228	0.6529
BLEU	0.5188	0.5884	0.3697	0.5459	0.4357	0.4607	0.4722
NIST	0.4218	0.4092	0.1721	0.3521	0.4769	0.3493	0.3326
NMG-WN	0.6658	0.6068	0.6116	0.6770	**0.5740**	0.5818	0.5669
METEOR	0.3881	0.4947	0.3127	0.2987	0.4162	0.3416	0.2958
WER	0.6656	0.6570	0.5740	0.7491	0.5301	0.6031	0.5205
Our method II	0.7676	0.7217	0.6343	0.7917	0.5474	0.6632	0.6774
BLEU with our method	0.6395	0.6696	0.5139	0.6611	0.5079	0.5698	0.5790

ues of the parameter are determined using English sentences from Reuters articles(Utiyama and Isahara, 2003). Moreover, we obtained the noun phrases using a shallow parser(Sha and Pereira, 2003) as the chunking tool. We revised some erroneous results that were obtained using the chunking tool.

3.2 Experimental Results

As described in this paper, we performed comparison experiments using our method and seven other methods. Tables 2 and 3 respectively show Pearson's correlation coefficient for sentence-level adequacy and fluency. Tables 4 and 5 respectively show Spearman's rank correlation coefficient for sentence-level adequacy and fluency. In Tables 2–5, bold typeface signifies the maximum correlation coefficients among eight automatic evaluation methods. Underlining in our method signifies that the differences between correlation coefficients obtained using our method and IMPACT are statistically significant at the 5% significance level. Moreover, "Avg." signifies the average of the correlation coefficients obtained by 12 machine translation systems in respective automatic evaluation methods, and "All" are the correlation coefficients using the scores of 1,200 output sentences obtained using the 12 machine translation systems.

3.3 Discussion

In Tables 2–5, the "Avg." score of our method is shown to be higher than those of other methods. Especially in terms of the sentence-level adequacy shown in Tables 2 and 4, "Avg." of our method is about 0.03 higher than that of IMPACT. Moreover, in system No. 8 and "All" of Tables 2 and 4, the differences between correlation coefficients obtained using our method and IMPACT are statistically significant at the 5% significance level.

Moreover, we investigated the correlation of machine translation systems of every type. Table 6 shows "All" of Pearson's correlation coefficient and Spearman's rank correlation coefficient in SMT (*i.e.*, system Nos. 1–2, system Nos. 4–8 and system Nos. 10–11) and RBMT (*i.e.*, system Nos. 3 and 12). The scores of 900 output sentences obtained by 9 machine

Table 3: Pearson's correlation coefficient for sentence-level fluency.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7
Our method	**0.5853**	**0.3782**	0.5689	0.4673	0.5739	**0.5344**	**0.7193**
IMPACT	0.5581	0.3407	0.5821	0.4586	**0.5768**	0.4852	0.6896
ROUGE-L	0.5551	0.3056	**0.5925**	0.4391	0.5666	0.4475	0.6756
BLEU	0.4793	0.0963	0.4488	0.3033	0.4690	0.3602	0.5272
NIST	0.4139	0.0257	0.4987	0.1682	0.3923	0.2236	0.3749
NMG-WN	0.5782	0.3090	0.5434	**0.4680**	0.5070	0.5234	0.5363
METEOR	0.4050	0.1405	0.4420	0.1825	0.4259	0.2336	0.4873
WER	0.5143	0.3031	0.5220	0.4262	0.4936	0.4405	0.6351
Our method II	0.5831	0.3689	0.5753	0.3991	0.5610	0.5445	0.7186
BLEU with our method	0.5425	0.2304	0.5115	0.3770	0.5358	0.4741	0.6142
	No. 8	No. 9	No. 10	No. 11	No. 12	Avg.	All
Our method	**0.5796**	**0.6424**	0.3241	0.5920	0.4321	**0.5331**	**0.5574**
IMPACT	0.5612	0.6320	0.3492	0.6034	0.4166	0.5211	0.5469
ROUGE-L	0.5414	0.6347	0.3231	0.5889	0.4127	0.5069	0.5387
BLEU	0.5040	0.5521	0.2134	0.4783	0.4078	0.4033	0.4278
NIST	0.3682	0.3811	0.1682	0.3116	**0.4484**	0.3146	0.3142
NMG-WN	0.5526	0.5799	**0.4509**	**0.6308**	0.4124	0.5007	0.5074
METEOR	0.2511	0.4153	0.1376	0.3351	0.2902	0.3122	0.2933
WER	0.5492	0.6421	0.3962	0.6228	0.4063	0.4960	0.4478
Our method II	0.5774	0.6486	0.3428	0.5975	0.4197	0.5280	0.5519
BLEU with our method	0.5660	0.6247	0.2536	0.5495	0.4550	0.4770	0.5014

translation systems in SMT and the scores of 200 output sentences obtained by 2 machine translation systems in RBMT are used respectively. However, EBMT is not included in Table 6 because EBMT is only system No. 9. In Table 6, our method obtained the highest correlation among the eight methods, except in terms of the adequacy of RBMT in Pearson's correlation coefficient. The differences between correlation coefficients obtained using our method and IMPACT are statistically significant at the 5% significance level for adequacy of SMT.

To confirm the effectiveness of noun-phrase chunking, we performed the experiment using a system combining BLEU with our method. In this case, BLEU scores were used as $score_{wd}$ in Eq. (13). This experimental result is shown as "BLEU with our method" in Tables 2–5. In the results of "BLEU with our method" in Tables 2–5, underlining signifies that the differences between correlation coefficients obtained using BLEU with our method and BLEU alone are statistically significant at the 5% significance level. The coefficients of correlation for BLEU with our method are higher than those of BLEU in any machine translation system, "Avg." and "All" in Tables 2–5. Moreover, for sentence-level adequacy, BLEU with our method is significantly better than BLEU in almost all machine translation systems and "All" in Tables 2 and 4. These results indicate that our method using noun-phrase chunking is effective for some methods and that it is statistically significant in each machine translation system, not only "All", which has large sentences.

Subsequently, we investigated the precision of the determination process of the corresponding noun phrases described in section 2.1: in the results of system No. 1, we calculated the precision as the ratio of the number of the correct corresponding noun phrases for the number of all noun-phrase correspondences obtained using the system based on our method. Results show that the precision was 93.4%, demonstrating that our method can determine the corresponding noun phrases correctly.

Moreover, we investigated the relation be-

Table 4: Spearman's rank correlation coefficient for sentence-level adequacy.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7
Our method	0.7456	**0.5049**	0.5837	**0.5146**	**0.6514**	**0.6557**	**0.6746**
IMPACT	0.7336	0.4881	0.5992	0.4741	0.6382	0.5841	0.6409
ROUGE-L	0.7304	0.4822	**0.6092**	0.4572	0.6135	0.5365	0.6368
BLEU	0.5525	0.2206	0.4327	0.3449	0.3230	0.2805	0.4375
NIST	0.5032	0.2438	0.4218	0.2489	0.2342	0.1534	0.3529
NMG-WN	**0.7541**	0.3829	0.5579	0.4472	0.5560	0.5828	0.6263
METEOR	0.4409	0.1509	0.4018	0.2580	0.3085	0.1991	0.4115
WER	0.6566	0.4147	0.5478	0.4272	0.5524	0.4884	0.5539
Our method II	0.7478	0.4972	0.5817	0.4892	0.6437	0.6428	0.6707
BLEU with our method	0.6644	0.3926	0.5065	0.4522	0.4639	0.4715	0.5460
	No. 8	No. 9	No. 10	No. 11	No. 12	Avg.	All
Our method	**0.7298**	**0.7258**	0.5961	**0.7633**	**0.6078**	**0.6461**	**0.6763**
IMPACT	0.6703	0.7067	0.5617	0.7411	0.5583	0.6164	0.6515
ROUGE-L	0.6603	0.6983	0.5340	0.7280	0.5281	0.6012	0.6435
BLEU	0.4571	0.5827	0.3220	0.4987	0.4302	0.4069	0.4227
NIST	0.4255	0.4424	0.1313	0.2950	0.4785	0.3276	0.3062
NMG-WN	0.6863	0.6524	**0.6412**	0.7015	0.5728	0.5968	0.5836
METEOR	0.4242	0.4776	0.3335	0.2861	0.4455	0.3448	0.2887
WER	0.6234	0.6480	0.5463	0.7131	0.5684	0.5617	0.4797
Our method II	0.7287	0.7255	0.5936	0.7761	0.5798	0.6397	0.6699
BLEU with our method	0.5850	0.6757	0.4596	0.6272	0.5452	0.5325	0.5474

tween the correlation obtained by our method and the quality of chunking. In "Our method" shown in Tables 2–5, noun phrases for which some erroneous results obtained using the chunking tool were revised. "Our method II" of Tables 2–5 used noun phrases that were given as results obtained using the chunking tool. Underlining in "Our method II" of Tables 2–5 signifies that the differences between correlation coefficients obtained using our method II and IMPACT are statistically significant at the 5% significance level. Fundamentally, in both "Avg." and "All" of Tables 2–5, the correlation coefficients of our method II without the revised noun phrases are lower than those of our method using the revised noun phrases. However, the difference between our method and our method II in "Avg." and "All" of Tables 2–5 is not large. The performance of the chunking tool has no great influence on the results of our method because $score_{wd}$ in Eqs. (3), (4), and (5) do not depend strongly on the performance of the chunking tool. For example, in sentences shown in Fig. 2, all common parts are the same as the common parts of Fig. 2 when "the crowning fall" in the MT output and "crowning drop" in the reference are not determined as the noun phrases. Other common parts are determined correctly because the weight of the common part "the amount of" is higher than those of other common parts by Eqs. (1) and (2). Consequently, the determination of the common parts except "the amount of" is not difficult.

In other language sentences, we already performed the experiments using Japanese sentences from Reuters articles(Oyamada et al., 2010). Results show that the correlation coefficients of IMPACT with our method, for which IMPACT scores were used as $score_{wd}$ in Eq. (13), were highest among some methods. Therefore, our method might not be language-dependent. Nevertheless, experiments using various language data are necessary to elucidate this point.

4 Conclusion

As described herein, we proposed a new automatic evaluation method for machine transla-

Table 5: Spearman's rank correlation coefficient for sentence-level fluency.

	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7
Our method	**0.5697**	0.3299	0.5446	0.4199	0.5733	0.5060	**0.6459**
IMPACT	0.5481	0.3285	0.5572	0.3976	**0.5960**	0.4317	0.6334
ROUGE-L	0.5470	0.3041	**0.5646**	0.3661	0.5638	0.3879	0.6255
BLEU	0.4157	0.0559	0.4286	0.2018	0.4475	0.2569	0.4909
NIST	0.4209	0.0185	0.4559	0.1093	0.3186	0.1898	0.3634
NMG-WN	0.5569	**0.3461**	0.5381	**0.4300**	0.5052	**0.5264**	0.5328
METEOR	0.4608	0.1429	0.4438	0.1783	0.4073	0.1596	0.4821
WER	0.4469	0.2395	0.5087	0.3292	0.4995	0.3482	0.5637
Our method II	0.5659	0.3216	0.5484	0.3773	0.5638	0.5211	0.6343
BLEU with our method	0.5188	0.1534	0.4793	0.3005	0.5255	0.3942	0.5676
	No. 8	No. 9	No. 10	No. 11	No. 12	Avg.	All
Our method	0.5646	**0.6617**	0.3319	0.6256	0.4485	**0.5185**	**0.5556**
IMPACT	0.5471	0.6454	0.3222	0.6319	0.4358	0.5062	0.5489
ROUGE-L	0.5246	0.6428	0.2949	0.6159	0.3928	0.4858	0.5359
BLEU	0.4882	0.5419	0.1407	0.4740	0.4176	0.3633	0.3971
NIST	0.4150	0.4193	0.0889	0.3006	**0.4752**	0.2980	0.2994
NMG-WN	**0.5684**	0.5850	**0.4451**	**0.6502**	0.4387	0.5102	0.5156
METEOR	0.2911	0.4267	0.1735	0.3264	0.3512	0.3158	0.2886
WER	0.5320	0.6505	0.3828	0.6501	0.4003	0.4626	0.4193
Our method II	0.5609	0.6687	0.3629	0.6223	0.4384	0.5155	0.5531
BLEU with our method	0.5470	0.6213	0.2184	0.5808	0.4870	0.4495	0.4825

Table 6: Correlation coefficient for SMT and RBMT.

	Pearson's correlation coefficient				Spearman's rank correlation coefficient			
	Adequacy		Fluency		Adequacy		Fluency	
	SMT	RBMT	SMT	RBMT	SMT	RBMT	SMT	RBMT
Our method	**0.7054**	0.5840	**0.5477**	**0.5016**	**0.6710**	**0.5961**	**0.5254**	**0.5003**
IMPACT	0.6721	0.5650	0.5364	0.4960	0.6397	0.5811	0.5162	0.4951
ROUGE-L	0.6560	0.5691	0.5179	0.4988	0.6225	0.5701	0.4942	0.4783
NMG-WN	0.5958	**0.5850**	0.5201	0.4732	0.6129	0.5755	0.5238	0.4959

tion. Our method calculates the scores for MT outputs using noun-phrase chunking. Consequently, the system obtains scores using the correctly matched words and phrase-level information based on the corresponding noun phrases. Experimental results demonstrate that our method yields the highest correlation among eight methods in terms of sentence-level adequacy and fluency.

Future studies will improve our method, enabling it to achieve high correlation in sentence-level fluency. Future studies will also include experiments using data of various languages.

Acknowledgements

This work was done as research under the AAMT/JAPIO Special Interest Group on Patent Translation. The Japan Patent Information Organization (JAPIO) and the National Institute of Informatics (NII) provided corpora used in this work. The author gratefully acknowledges JAPIO and NII for their support. Moreover, this work was partially supported by Grants from the High-Tech Research Center of Hokkai-Gakuen University and the Kayamori Foundation of Informational Science Advancement.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *In Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72.

Deborah Coughlin. 2003. Correlating Automated and Human Assessments of Machine Translation Quality. *In Proc. of MT Summit IX*, 63–70.

Hiroshi Echizen-ya and Kenji Araki. 2007. Automatic Evaluation of Machine Translation based on Recursive Acquisition of an Intuitive Common Parts Continuum. *In Proc. of MT Summit XII*, 151–158.

Hiroshi Echizen-ya, Terumasa Ehara, Sayori Shimohata, Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro and Noriko Kando. 2009. Meta-Evaluation of Automatic Evaluation Methods for Machine Translation using Patent Translation Data in NTCIR-7. *In Proc. of the 3rd Workshop on Patent Translation*, 9–16.

Terumasa Ehara. 2007. Rule Based Machine Translation Combined with Statistical Post Editor for Japanese to English Patent Translation. *In Proc. of MT Summit XII Workshop on Patent Translation*, 13–18.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto and Takehito Utsuro. 2008. Overview of the Patent Translation Task at the NTCIR-7 Workshop. *In Proc. of 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, 389–400.

Gregor Leusch, Nicola Ueffing and Hermann Ney. 2003. A Novel String-to-String Distance Measure with Applications to Machine Translation Evaluation. *In Proc. of MT Summit IX*, 240–247.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. *In Proc. of ACL'04*, 606–613.

Dennis N. Mehay and Chris Brew. 2007. BLEUÂTRE: Flattening Syntactic Dependencies for MT Evaluation. *In Proc. of MT Summit XII*, 122–131.

Andrew Mutton, Mark Dras, Stephen Wan and Robert Dale. 2007. GLEU: Automatic Evaluation of Sentence-Level Fluency. *In Proc. of ACL'07*, 344–351.

NIST. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. *http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf*.

Takashi Oyamada, Hiroshi Echizen-ya and Kenji Araki. 2010. Automatic Evaluation of Machine Translation Using both Words Information and Comprehensive Phrases Information. *In IPSJ SIG Technical Report, Vol.2010-NL-195, No. 3 (in Japanese)*.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *In Proc. of ACL'02*, 311–318.

Michael Pozar and Eugene Charniak. 2006. Bllip: An Improved Evaluation Metric for Machine Translation. *Brown University Master Thesis*.

Fei Sha and Fernando Pereira. 2003. Shallow Parsing with Conditional Random Fields. *In Proc. of HLT-NAACL 2003*, 134–141.

Keh-Yih Su, Ming-Wen Wu and Jing-Shin Chang. 1992. A New Quantitative Quality Measure for Machine Translation Systems. *In Proc. of GOLING'92*, 433–439.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable Measures for Aligning Japanese–English News Articles and Sentences. *In Proc. of the ACL'03*, pp.72–79.

Open Information Extraction using Wikipedia

Fei Wu
University of Washington
Seattle, WA, USA
wufei@cs.washington.edu

Daniel S. Weld
University of Washington
Seattle, WA, USA
weld@cs.washington.edu

Abstract

Information-extraction (IE) systems seek to distill semantic relations from natural-language text, but most systems use supervised learning of relation-specific examples and are thus limited by the availability of training data. *Open* IE systems such as TextRunner, on the other hand, aim to handle the unbounded number of relations found on the Web. But how well can these open systems perform?

This paper presents WOE, an open IE system which improves dramatically on TextRunner's precision and recall. The key to WOE's performance is a novel form of self-supervised learning for open extractors — using heuristic matches between Wikipedia infobox attribute values and corresponding sentences to construct training data. Like TextRunner, WOE's extractor eschews lexicalized features and handles an unbounded set of semantic relations. WOE can operate in two modes: when restricted to POS tag features, it runs as quickly as TextRunner, but when set to use dependency-parse features its precision and recall rise even higher.

1 Introduction

The problem of information-extraction (IE), generating relational data from natural-language text, has received increasing attention in recent years. A large, high-quality repository of extracted tuples can potentially benefit a wide range of NLP tasks such as question answering, ontology learning, and summarization. The vast majority of IE work uses supervised learning of relation-specific examples. For example, the WebKB project (Craven et al., 1998) used labeled examples of the `courses-taught-by` relation to induce rules for identifying additional instances of the relation. While these methods can achieve high precision and recall, they are limited by the availability of training data and are unlikely to scale to the thousands of relations found in text on the Web.

An alternative paradigm, *Open* IE, pioneered by the TextRunner system (Banko et al., 2007) and the "preemptive IE" in (Shinyama and Sekine, 2006), aims to handle an unbounded number of relations and run quickly enough to process Web-scale corpora. Domain independence is achieved by extracting the relation name as well as its two arguments. Most open IE systems use self-supervised learning, in which automatic heuristics generate labeled data for training the extractor. For example, TextRunner uses a small set of hand-written rules to heuristically label training examples from sentences in the Penn Treebank.

This paper presents WOE (Wikipedia-based Open Extractor), the first system that autonomously transfers knowledge from random editors' effort of collaboratively editing Wikipedia to train an open information extractor. Specifically, WOE generates *relation-specific* training examples by matching *Infobox*[1] attribute values to corresponding sentences (as done in Kylin (Wu and Weld, 2007) and Luchs (Hoffmann et al., 2010)), but WOE abstracts these examples to *relation-independent* training data to learn an unlexicalized extractor, akin to that of TextRunner. WOE can operate in two modes: when restricted to shallow features like part-of-speech (POS) tags, it runs as quickly as Textrunner, but when set to use dependency-parse features its precision and recall rise even higher. We present a thorough experimental evaluation, making the following contributions:

- We present WOE, a new approach to open IE that uses Wikipedia for self-supervised learn-

[1] An infobox is a set of tuples summarizing the key attributes of the subject in a Wikipedia article. For example, the infobox in the article on "Sweden" contains attributes like *Capital, Population* and *GDP*.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 118–127,
Uppsala, Sweden, 11-16 July 2010.

ing of unlexicalized extractors. Compared with TextRunner (the state of the art) on three corpora, WOE yields between 72% and 91% improved F-measure — generalizing well beyond Wikipedia.

- Using the same learning algorithm and features as TextRunner, we compare four different ways to generate positive and negative training data with TextRunner's method, concluding that our Wikipedia heuristic is responsible for the bulk of WOE's improved accuracy.
- The biggest win arises from using parser features. Previous work (Jiang and Zhai, 2007) concluded that parser-based features are unnecessary for information extraction, but that work assumed the presence of lexical features. We show that abstract dependency paths are a highly informative feature when performing unlexicalized extraction.

2 Problem Definition

An open information extractor is a function from a document, d, to a set of triples, $\{\langle \texttt{arg}_1, \texttt{rel}, \texttt{arg}_2 \rangle\}$, where the args are noun phrases and rel is a textual fragment indicating an implicit, semantic relation between the two noun phrases. The extractor should produce one triple for every relation stated *explicitly* in the text, but is not required to infer implicit facts. In this paper, we assume that all relational instances are stated within a single sentence. Note the difference between open IE and the traditional approaches (*e.g.*, as in WebKB), where the task is to decide whether some pre-defined relation holds between (two) arguments in the sentence.

We wish to learn an open extractor *without direct supervision*, *i.e.* without annotated training examples or hand-crafted patterns. Our input is Wikipedia, a collaboratively-constructed encyclopedia[2]. As output, WOE produces an unlexicalized and relation-independent open extractor. Our objective is an extractor which generalizes beyond Wikipedia, handling other corpora such as the general Web.

3 Wikipedia-based Open IE

The key idea underlying WOE is the automatic construction of training examples by heuristically matching Wikipedia infobox values and corresponding text; these examples are used to generate an unlexicalized, relation-independent (open) extractor. As shown in Figure 1, WOE has three main components: preprocessor, matcher, and learner.

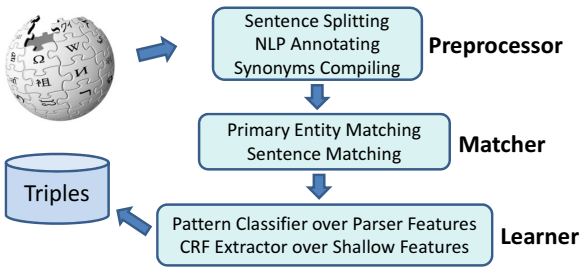


Figure 1: Architecture of WOE.

3.1 Preprocessor

The preprocessor converts the raw Wikipedia text into a sequence of sentences, attaches NLP annotations, and builds synonym sets for key entities. The resulting data is fed to the matcher, described in Section 3.2, which generates the training set.

Sentence Splitting: The preprocessor first renders each Wikipedia article into HTML, then splits the article into sentences using OpenNLP.

NLP Annotation: As we discuss fully in Section 4 (Experiments), we consider several variations of our system; one version, WOE^{parse}, uses parser-based features, while another, WOE^{pos}, uses shallow features like POS tags, which may be more quickly computed. Depending on which version is being trained, the preprocessor uses OpenNLP to supply POS tags and NP-chunk annotations — or uses the Stanford Parser to create a dependency parse. When parsing, we force the hyperlinked anchor texts to be a single token by connecting the words with an underscore; this transformation improves parsing performance in many cases.

Compiling Synonyms: As a final step, the preprocessor builds sets of synonyms to help the matcher find sentences that correspond to infobox relations. This is useful because Wikipedia editors frequently use multiple names for an entity; for example, in the article titled "University of Washington" the token "UW" is widely used to refer the university. Additionally, attribute values are often described differently within the infobox than they are in surrounding text. Without knowledge of these synonyms, it is impossible to construct good matches. Following (Wu and Weld, 2007; Nakayama and Nishio, 2008), the preprocessor uses Wikipedia redirection pages and back-

[2]We also use DBpedia (Auer and Lehmann, 2007) as a collection of conveniently parsed Wikipedia *infoboxes*

ward links to automatically construct synonym sets. Redirection pages are a natural choice, because they explicitly encode synonyms; for example, "USA" is redirected to the article on the "United States." Backward links for a Wikipedia entity such as the "Massachusetts Institute of Technology" are hyperlinks pointing to this entity from other articles; the anchor text of such links (*e.g.*, "MIT") forms another source of synonyms.

3.2 Matcher

The matcher constructs training data for the learner component by heuristically matching attribute-value pairs from Wikipedia articles containing infoboxes with corresponding sentences in the article. Given the article on "Stanford University," for example, the matcher should associate $\langle \texttt{established}, 1891 \rangle$ with the sentence "The university was founded in 1891 by ..." Given a Wikipedia page with an infobox, the matcher iterates through all its attributes looking for a unique sentence that contains references to both the subject of the article and the attribute value; these noun phrases will be annotated $\texttt{arg}_1$ and $\texttt{arg}_2$ in the training set. The matcher considers a sentence to contain the attribute value if the value or its synonym is present. Matching the article subject, however, is more involved.

Matching Primary Entities: In order to match shorthand terms like "MIT" with more complete names, the matcher uses an ordered set of heuristics like those of (Wu and Weld, 2007; Nguyen et al., 2007):

- Full match: strings matching the full name of the entity are selected.
- Synonym set match: strings appearing in the entity's synonym set are selected.
- Partial match: strings matching a prefix or suffix of the entity's name are selected. If the full name contains punctuation, only a prefix is allowed. For example, "Amherst" matches "Amherst, Mass," but "Mass" does not.
- Patterns of "the <*type*>": The matcher first identifies the type of the entity (*e.g.*, "city" for "Ithaca"), then instantiates the pattern to create the string "the city." Since the first sentence of most Wikipedia articles is stylized (*e.g.* "The city of Ithaca sits ..."), a few patterns suffice to extract most entity types.
- The most frequent pronoun: The matcher assumes that the article's most frequent pronoun denotes the primary entity, *e.g.*, "he" for the page on "Albert Einstein." This heuristic is dropped when "it" is most common, because the word is used in too many other ways.

When there are multiple matches to the primary entity in a sentence, the matcher picks the one which is closest to the matched infobox attribute value in the parser dependency graph.

Matching Sentences: The matcher seeks a *unique* sentence to match the attribute value. To produce the best training set, the matcher performs three filterings. First, it skips the attribute completely when multiple sentences mention the value or its synonym. Second, it rejects the sentence if the subject and/or attribute value are not heads of the noun phrases containing them. Third, it discards the sentence if the subject and the attribute value do not appear in the same clause (or in parent/child clauses) in the parse tree.

Since Wikipedia's Wikimarkup language is semantically ambiguous, parsing infoboxes is surprisingly complex. Fortunately, DBpedia (Auer and Lehmann, 2007) provides a cleaned set of infoboxes from 1,027,744 articles. The matcher uses this data for attribute values, generating a training dataset with a total of 301,962 labeled sentences.

3.3 Learning Extractors

We learn two kinds of extractors, one (WOE^{parse}) using features from dependency-parse trees and the other (WOE^{pos}) limited to shallow features like POS tags. WOE^{parse} uses a pattern learner to classify whether the shortest dependency path between two noun phrases indicates a semantic relation. In contrast, WOE^{pos} (like TextRunner) trains a conditional random field (CRF) to output certain text between noun phrases when the text denotes such a relation. Neither extractor uses individual words or lexical information for features.

3.3.1 Extraction with Parser Features

Despite some evidence that parser-based features have limited utility in IE (Jiang and Zhai, 2007), we hoped dependency paths would improve precision on long sentences.

Shortest Dependency Path as Relation: Unless otherwise noted, WOE uses the Stanford Parser to create dependencies in the "collapsedDependency" format. Dependencies involving prepositions, conjuncts as well as information about the referent of relative clauses are collapsed to get direct dependencies between content words. As

noted in (de Marneffe and Manning, 2008), this collapsed format often yields simplified patterns which are useful for relation extraction. Consider the sentence:

Dan was not born in Berkeley.

The Stanford Parser dependencies are:

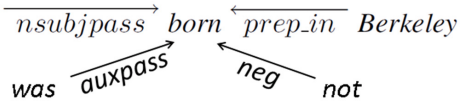
nsubjpass(born-4, Dan-1)
auxpass(born-4, was-2)
neg(born-4, not-3)
prep_in(born-4, Berkeley-6)

where each atomic formula represents a binary dependence from dependent token to the governor token.

These dependencies form a directed graph, $\langle V, E \rangle$, where each token is a vertex in V, and E is the set of dependencies. For any pair of tokens, such as "Dan" and "Berkeley", we use the shortest connecting path to represent the possible relation between them:

Dan $\overrightarrow{nsubjpass}$ *born* $\overleftarrow{prep\_in}$ *Berkeley*

We call such a path a *corePath*. While we will see that corePaths are useful for indicating *when* a relation exists between tokens, they don't necessarily capture the *semantics* of that relation. For example, the path shown above doesn't indicate the existence of negation! In order to capture the *meaning* of the relation, the learner augments the corePath into a tree by adding all adverbial and adjectival modifiers as well as dependencies like "neg" and "auxpass". We call the result an *expandPath* as shown below:



WOE traverses the expandPath with respect to the token orders in the original sentence when *outputting* the final expression of `rel`.

Building a Database of Patterns: For each of the 301,962 sentences selected and annotated by the matcher, the learner generates a corePath between the tokens denoting the subject and the infobox attribute value. Since we are interested in eventually extracting "subject, relation, object" triples, the learner rejects corePaths that don't start with subject-like dependencies, such as *nsubj*, *nsubjpass*, *partmod* and *rcmod*. This leads to a collection of 259,046 corePaths.

To combat data sparsity and improve learning performance, the learner further generalizes the corePaths in this set to create a smaller set of *generalized-corePaths*. The idea is to eliminate distinctions which are irrelevant for recognizing (domain-independent) relations. Lexical words in corePaths are replaced with their POS tags. Further, all Noun POS tags and "PRP" are abstracted to "N", all Verb POS tags to "V", all Adverb POS tags to "RB" and all Adjective POS tags to "J". The preposition dependencies such as "prep_in" are generalized to "prep". Take the corePath "*Dan* $\overrightarrow{nsubjpass}$ *born* $\overleftarrow{prep\_in}$ *Berkeley*" for example, its generalized-corePath is "*N* $\overrightarrow{nsubjpass}$ *V* $\overleftarrow{prep}$ *N*". We call such a generalized-corePath an extraction pattern. In total, WOE builds a database (named DB_p) of 15,333 distinct patterns and each pattern p is associated with a frequency — the number of matching sentences containing p. Specifically, 185 patterns have $f_p \geq 100$ and 1929 patterns have $f_p \geq 5$.

Learning a Pattern Classifier: Given the large number of patterns in DB_p, we assume few valid open extraction patterns are left behind. The learner builds a simple pattern classifier, named WOE^{parse}, which checks whether the generalized-corePath from a test triple is present in DB_p, and computes the normalized logarithmic frequency as the probability[3]:

$$w(p) = \frac{max(log(f_p) - log(f_{min}), 0)}{log(f_{max}) - log(f_{min})}$$

where f_{max} (50,259 in this paper) is the maximal frequency of pattern in DB_p, and f_{min} (set 1 in this work) is the controlling threshold that determines the minimal frequency of a valid pattern.

Take the previous sentence "*Dan was not born in Berkeley*" for example. WOE^{parse} first identifies *Dan* as $\texttt{arg}_1$ and *Berkeley* as $\texttt{arg}_2$ based on NP-chunking. It then computes the corePath "*Dan* $\overrightarrow{nsubjpass}$ *born* $\overleftarrow{prep\_in}$ *Berkeley*" and abstracts to *p*="*N* $\overrightarrow{nsubjpass}$ *V* $\overleftarrow{prep}$ *N*". It then queries DB_p to retrieve the frequency $f_p = 29112$ and assigns a probability of 0.95. Finally, WOE^{parse} traverses the triple's expandPath to output the final expression $\langle Dan, wasNotBornIn, Berkeley \rangle$. As shown in the experiments on three corpora, WOE^{parse} achieves an F-measure which is between 72% to 91% greater than TextRunner's.

3.3.2 Extraction with Shallow Features

WOE^{parse} has a dramatic performance improvement over TextRunner. However, the improvement comes at the cost of speed — TextRunner

[3]How to learn a more sophisticated weighting function is left as a future topic.

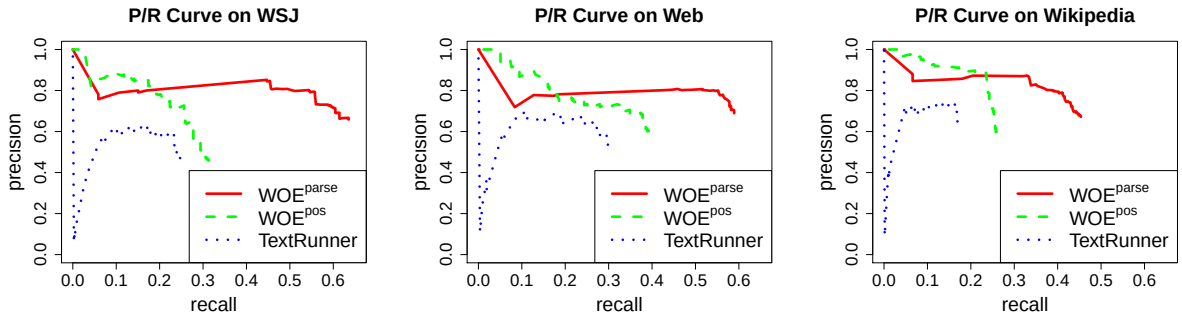


Figure 2: WOE^{pos} performs better than TextRunner, especially on precision. WOE^{parse} dramatically improves performance, especially on recall.

runs about $30X$ faster by only using shallow features. Since high speed can be crucial when processing Web-scale corpora, we additionally learn a CRF extractor WOE^{pos} based on shallow features like POS-tags. In both cases, however, we generate training data from Wikipedia by matching sentences with infoboxes, while TextRunner used a small set of hand-written rules to label training examples from the Penn Treebank.

We use the same matching sentence set behind DB_p to generate positive examples for WOE^{pos}. Specifically, for each matching sentence, we label the subject and infobox attribute value as $\texttt{arg}_1$ and $\texttt{arg}_2$ to serve as the ends of a linear CRF chain. Tokens involved in the expandPath are labeled as `rel`. Negative examples are generated from random noun-phrase pairs in other sentences when their generalized-CorePaths are not in DB_p.

WOE^{pos} uses the same learning algorithm and selection of features as TextRunner: a two-order CRF chain model is trained with the Mallet package (McCallum, 2002). WOE^{pos}'s features include POS-tags, regular expressions (*e.g.*, for detecting capitalization, punctuation, *etc.*.), and conjunctions of features occurring in adjacent positions within six words to the left and to the right of the current word.

As shown in the experiments, WOE^{pos} achieves an improved F-measure over TextRunner between 18% to 34% on three corpora, and this is mainly due to the increase on precision.

4 Experiments

We used three corpora for experiments: WSJ from Penn Treebank, Wikipedia, and the general Web. For each dataset, we randomly selected 300 sentences. Each sentence was examined by two people to label all reasonable triples. These candidate triples are mixed with pseudo-negative ones and submitted to Amazon Mechanical Turk for verification. Each triple was examined by 5 Turkers. We mark a triple's final label as positive when more than 3 Turkers marked them as positive.

4.1 Overall Performance Analysis

In this section, we compare the overall performance of WOE^{parse}, WOE^{pos} and TextRunner (shared by the Turing Center at the University of Washington). In particular, we are going to answer the following questions: 1) How do these systems perform against each other? 2) How does performance vary w.r.t. sentence length? 3) How does extraction speed vary w.r.t. sentence length?

Overall Performance Comparison

The detailed P/R curves are shown in Figure 2. To have a close look, for each corpus, we randomly divided the 300 sentences into 5 groups and compared the best F-measures of three systems in Figure 3. We can see that:

- WOE^{pos} is better than TextRunner, especially on precision. This is due to better training data from Wikipedia via self-supervision. Section 4.2 discusses this in more detail.
- WOE^{parse} achieves the best performance, especially on recall. This is because the parser features help to handle complicated and long-distance relations in difficult sentences. In particular, WOE^{parse} outputs 1.42 triples per sentence on average, while WOE^{pos} outputs 1.05 and TextRunner outputs 0.75.

Note that we measure TextRunner's precision & recall differently than (Banko et al., 2007) did. Specifically, we compute the precision & recall based on *all* extractions, while Banko et al. counted only *concrete* triples where $\texttt{arg}_1$ is a proper noun, $\texttt{arg}_2$ is a proper noun or date, and

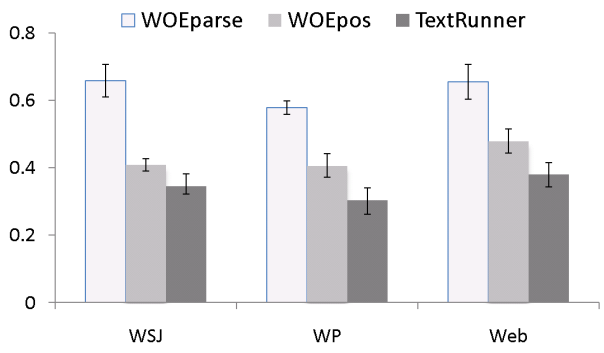


Figure 3: WOE^{pos}achieves an F-measure, which is between 18% and 34% better than TextRunner's. WOE^{parse}achieves an improvement between 72% and 91% over TextRunner. The error bar indicates one standard deviation.

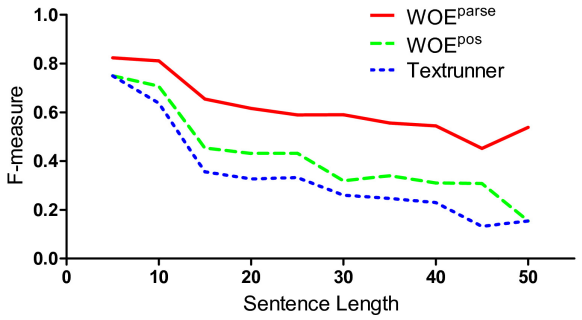


Figure 4: WOE^{parse}'s F-measure decreases more slowly with sentence length than WOE^{pos} and TextRunner, due to its better handling of difficult sentences using parser features.

the frequency of `rel` is over a threshold. Our experiments show that focussing on concrete triples generally improves precision at the expense of recall.[4] Of course, one can apply a concreteness filter to any open extractor in order to trade recall for precision.

The extraction errors by WOE^{parse} can be categorized into four classes. We illustrate them with the WSJ corpus. In total, WOE^{parse} got 85 wrong extractions on WSJ, and they are caused by: 1) Incorrect $\texttt{arg}_1$ and/or $\texttt{arg}_2$ from NP-Chunking (18.6%); 2) A erroneous dependency parse from Stanford Parser (11.9%); 3) Inaccurate meaning (27.1%) — for example, $\langle she, isNominatedBy, PresidentBush\rangle$ is wrongly extracted from the sentence "If she is nominated by President Bush ..."[5]; 4) A pattern inapplicable for the test sentence (42.4%).

Note WOE^{parse} is worse than WOE^{pos} in the low recall region. This is mainly due to parsing errors (especially on long-distance dependencies), which misleads WOE^{parse} to extract false high-confidence triples. WOE^{pos} won't suffer from such parsing errors. Therefore it has better precision on high-confidence extractions.

We noticed that TextRunner has a dip point in the low recall region. There are two typical errors responsible for this. A sample error of the first type is $\langle Sources, sold, theCompany\rangle$ extracted from the sentence "Sources said he sold the company", where "Sources" is wrongly treated as the subject of the object clause. A sample error of the second type is $\langle thisYear, willStarIn, theMovie\rangle$ extracted from the sentence "Coming up this year, Long will star in the new movie.", where "this year" is wrongly treated as part of a compound subject. Taking the WSJ corpus for example, at the dip point with recall=0.002 and precision=0.059, these two types of errors account for 70% of all errors.

Extraction Performance *vs.* **Sentence Length**

We tested how extractors' performance varies with sentence length; the results are shown in Figure 4. TextRunner and WOE^{pos} have good performance on short sentences, but their performance deteriorates quickly as sentences get longer. This is because long sentences tend to have complicated and long-distance relations which are difficult for shallow features to capture. In contrast, WOE^{parse}'s performance decreases more slowly w.r.t. sentence length. This is mainly because parser features are more useful for handling difficult sentences and they help WOE^{parse} to maintain a good recall with only moderate loss of precision.

Extraction Speed *vs.* **Sentence Length**

We also tested the extraction speed of different extractors. We used Java for implementing the extractors, and tested on a Linux platform with a 2.4GHz CPU and 4G memory. On average, it takes WOE^{parse} 0.679 seconds to process a sentence. For TextRunner and WOE^{pos}, it only takes 0.022 seconds — $30X$ times faster. The detailed extraction speed *vs.* sentence length is in Figure 5, showing that TextRunner and WOE^{pos}'s extraction time grows approximately linearly with sentence length, while WOE^{parse}'s extraction time grows

[4]For example, consider the Wikipedia corpus. From our 300 test sentences, TextRunner extracted 257 triples (at 72.0% precision) but only extracted 16 concrete triples (with 87.5% precision).

[5]These kind of errors might be excluded by monitoring whether sentences contain words such as 'if,' 'suspect,' 'doubt,' *etc.*. We leave this as a topic for the future.

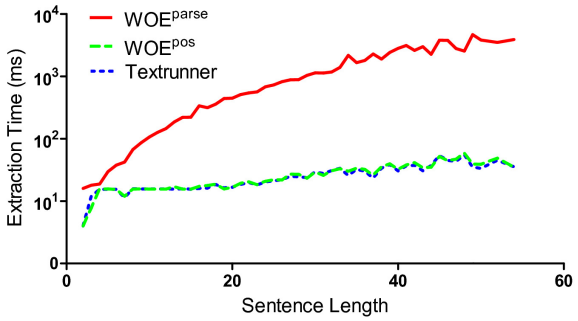


Figure 5: Textrnner and WOE^{pos}'s running time seems to grow linearly with sentence length, while WOE^{parse}'s time grows quadratically.

quadratically ($R^2 = 0.935$) due to its reliance on parsing.

4.2 Self-supervision with Wikipedia Results in Better Training Data

In this section, we consider how the process of matching Wikipedia infobox values to corresponding sentences results in better training data than the hand-written rules used by TextRunner.

To compare with TextRunner, we tested four different ways to generate training examples from Wikipedia for learning a CRF extractor. Specifically, positive and/or negative examples are selected by TextRunner's hand-written rules (*tr* for short), by WOE's heuristic of matching sentences with infoboxes (*w* for short), or randomly (*r* for short). We use CRF$_{+h_1-h_2}$ to denote a particular approach, where "+" means positive samples, "-" means negative samples, and $h_i \in \{tr, w, r\}$. In particular, "$+w$" results in 221,205 positive examples based on the matching sentence set[6]. All extractors are trained using about the same number of positive and negative examples. In contrast, TextRunner was trained with 91,687 positive examples and 96,795 negative examples generated from the WSJ dataset in Penn Treebank.

The CRF extractors are trained using the same learning algorithm and feature selection as TextRunner. The detailed P/R curves are in Figure 6, showing that using WOE heuristics to label positive examples gives the biggest performance boost. CRF$_{+tr-tr}$ (trained using TextRunner's heuristics) is slightly worse than TextRunner. Most likely, this is because TextRunner's heuristics rely on parse trees to label training examples, and the Stanford parse on Wikipedia is less accurate than the gold parse on WSJ.

[6]This number is smaller than the total number of corePaths (259,046) because we require $\texttt{arg}_1$ to appear before $\texttt{arg}_2$ in a sentence — as specified by TextRunner.

4.3 Design Desiderata of WOE^{parse}

There are two interesting design choices in WOE^{parse}: 1) whether to require $\texttt{arg}_1$ to appear before $\texttt{arg}_2$ (denoted as $1 \prec 2$) in the sentence; 2) whether to allow corePaths to contain prepositional phrase (PP) attachments (denoted as *PPa*). We tested how they affect the extraction performance; the results are shown in Figure 7.

We can see that filtering PP attachments ($\overline{PPa}$) gives a large precision boost with a noticeable loss in recall; enforcing a lexical ordering of relation arguments ($1 \prec 2$) yields a smaller improvement in precision with small loss in recall. Take the WSJ corpus for example: setting $1 \prec 2$ and $\overline{PPa}$ achieves a precision of 0.792 (with recall of 0.558). By changing $1 \prec 2$ to $1 \sim 2$, the precision decreases to 0.773 (with recall of 0.595). By changing $\overline{PPa}$ to *PPa* and keeping $1 \prec 2$, the precision decreases to 0.642 (with recall of 0.687) — in particular, if we use gold parse, the precision decreases to 0.672 (with recall of 0.685). We set $1 \prec 2$ and $\overline{PPa}$ as default in WOE^{parse} as a logical consequence of our preference for high precision over high recall.

4.3.1 Different parsing options

We also tested how different parsing might effect WOE^{parse}'s performance. We used three parsing options on the WSJ dataset: Stanford parsing, CJ50 parsing (Charniak and Johnson, 2005), and the gold parses from the Penn Treebank. The Stanford Parser is used to derive dependencies from CJ50 and gold parse trees. Figure 8 shows the detailed P/R curves. We can see that although today's statistical parsers make errors, they have negligible effect on the accuracy of WOE.

5 Related Work

Open or Traditional Information Extraction: Most existing work on IE is relation-specific. Occurrence-statistical models (Agichtein and Gravano, 2000; M. Ciaramita, 2005), graphical models (Peng and McCallum, 2004; Poon and Domingos, 2008), and kernel-based methods (Bunescu and R.Mooney, 2005) have been studied. Snow et al. (Snow et al., 2005) utilize WordNet to learn dependency path patterns for extracting the hypernym relation from text. Some seed-based frameworks are proposed for open-domain extraction (Pasca, 2008; Davidov et al., 2007; Davidov and Rappoport, 2008). These works focus

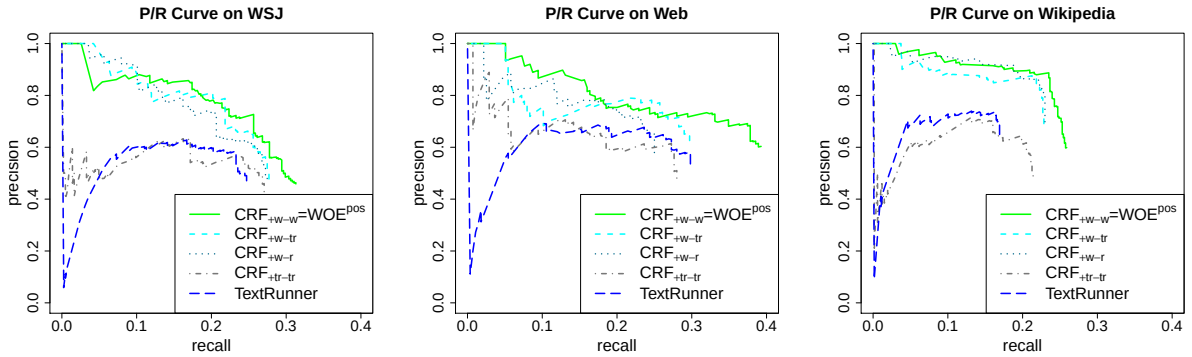


Figure 6: Matching sentences with Wikipedia infoboxes results in better training data than the hand-written rules used by TextRunner.

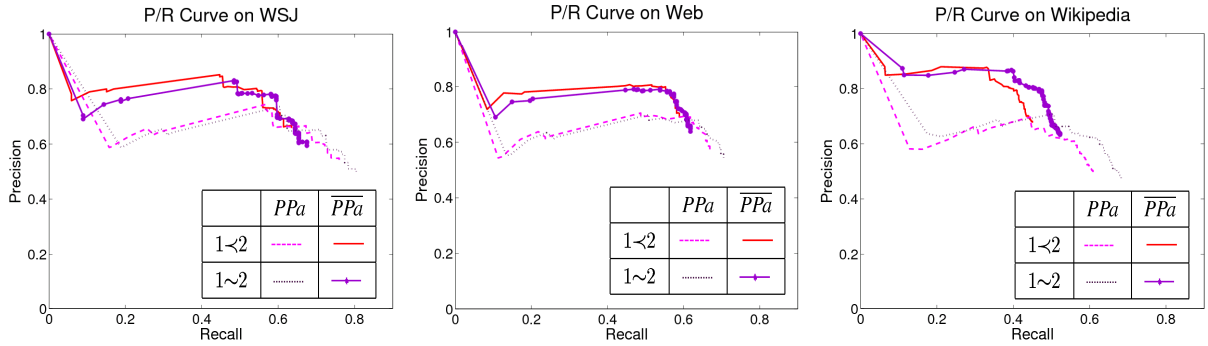


Figure 7: Filtering prepositional phrase attachments ($\overline{PPa}$) shows a strong boost to precision, and we see a smaller boost from enforcing a lexical ordering of relation arguments ($1 \prec 2$).

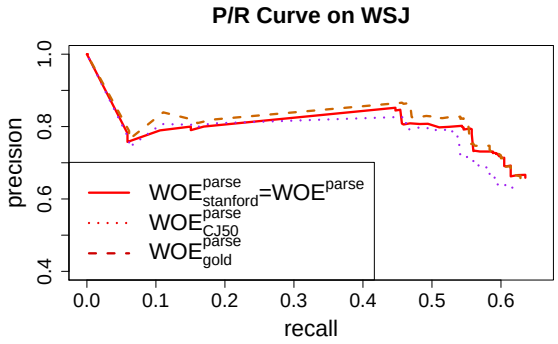


Figure 8: Although today's statistical parsers make errors, they have negligible effect on the accuracy of WOE compared to operation on gold standard, human-annotated data.

on identifying general relations such as class attributes, while open IE aims to extract relation instances from given sentences. Another seed-based system StatSnowball (Zhu et al., 2009) can perform both relation-specific and open IE by iteratively generating weighted extraction patterns. Different from WOE, StatSnowball only employs shallow features and uses L1-normalization to weight patterns. Shinyama and Sekine proposed the "preemptive IE" framework to avoid relation-specificity (Shinyama and Sekine, 2006). They first group documents based on pairwise vector-space clustering, then apply an additional clustering to group entities based on documents clusters. The two clustering steps make it difficult to meet the scalability requirement necessary to process the Web. Mintz et al. (Mintz et al., 2009) uses Freebase to provide distant supervision for relation extraction. They applied a similar heuristic by matching Freebase tuples with unstructured sentences (Wikipedia articles in their experiments) to create features for learning relation extractors. Matching Freebase with arbitrary sentences instead of matching Wikipedia infobox with corresponding Wikipedia articles will potentially increase the size of matched sentences at a cost of accuracy. Also, their learned extractors are relation-specific. Alan Akbik et al. (Akbik and Broß, 2009) annotated 10,000 sentences parsed with LinkGrammar and selected 46 general linkpaths as patterns for relation extraction. In contrast, WOE learns 15,333 general patterns based on an automatically annotated set of

301,962 Wikipedia sentences. The KNext system (Durme and Schubert, 2008) performs open knowledge extraction via significant heuristics. Its output is knowledge represented as logical statements instead of information represented as segmented text fragments.

Information Extraction with Wikipedia: The YAGO system (Suchanek et al., 2007) extends WordNet using facts extracted from Wikipedia categories. It only targets a limited number of predefined relations. Nakayama et al. (Nakayama and Nishio, 2008) parse selected Wikipedia sentences and perform extraction over the phrase structure trees based on several handcrafted patterns. Wu and Weld proposed the KYLIN system (Wu and Weld, 2007; Wu et al., 2008) which has the same spirit of matching Wikipedia sentences with infoboxes to learn CRF extractors. However, it only works for relations defined in Wikipedia infoboxes.

Shallow or Deep Parsing: Shallow features, like POS tags, enable fast extraction over large-scale corpora (Davidov et al., 2007; Banko et al., 2007). Deep features are derived from parse trees with the hope of training better extractors (Zhang et al., 2006; Zhao and Grishman, 2005; Bunescu and Mooney, 2005; Wang, 2008). Jiang and Zhai (Jiang and Zhai, 2007) did a systematic exploration of the feature space for relation extraction on the ACE corpus. Their results showed limited advantage of parser features over shallow features for IE. However, our results imply that abstracted dependency path features are highly informative for open IE. There might be several reasons for the different observations. First, Jiang and Zhai's results are tested for traditional IE where local lexicalized tokens might contain sufficient information to trigger a correct classification. The situation is different when features are completely unlexicalized in open IE. Second, as they noted, many relations defined in the ACE corpus are short-range relations which are easier for shallow features to capture. In practical corpora like the general Web, many sentences contain complicated long-distance relations. As we have shown experimentally, parser features are more powerful in handling such cases.

6 Conclusion

This paper introduces WOE, a new approach to open IE that uses self-supervised learning over unlexicalized features, based on a heuristic match between Wikipedia infoboxes and corresponding text. WOE can run in two modes: a CRF extractor (WOE^{pos}) trained with shallow features like POS tags; a pattern classfier (WOE^{parse}) learned from dependency path patterns. Comparing with TextRunner, WOE^{pos} runs at the same speed, but achieves an F-measure which is between 18% and 34% greater on three corpora; WOE^{parse} achieves an F-measure which is between 72% and 91% higher than that of TextRunner, but runs about $30X$ times slower due to the time required for parsing.

Our experiments uncovered two sources of WOE's strong performance: 1) the Wikipedia heuristic is responsible for the bulk of WOE's improved accuracy, but 2) dependency-parse features are highly informative when performing unlexicalized extraction. We note that this second conclusion disagrees with the findings in (Jiang and Zhai, 2007).

In the future, we plan to run WOE over the billion document CMU ClueWeb09 corpus to compile a giant knowledge base for distribution to the NLP community. There are several ways to further improve WOE's performance. Other data sources, such as Freebase, could be used to create an additional training dataset via self-supervision. For example, Mintz et al. consider all sentences containing both the subject and object of a Freebase record as matching sentences (Mintz et al., 2009); while they use this data to learn relation-specific extractors, one could also learn an open extractor. We are also interested in merging lexicalized and open extraction methods; the use of some domain-specific lexical features might help to improve WOE's practical performance, but the best way to do this is unclear. Finally, we wish to combine WOE^{parse} with WOE^{pos} (*e.g.*, with voting) to produce a system which maximizes precision at low recall.

Acknowledgements

We thank Oren Etzioni and Michele Banko from Turing Center at the University of Washington for providing the code of their software and useful discussions. We also thank Alan Ritter, Mausam, Peng Dai, Raphael Hoffmann, Xiao Ling, Stefan Schoenmackers, Andrey Kolobov and Daniel Suskin for valuable comments. This material is based upon work supported by the WRF / TJ Cable Professorship, a gift from Google and by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions,

findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL).

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *ICDL*.

Alan Akbik and Jügen Broß. 2009. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In *WWW Workshop*.

Sören Auer and Jens Lehmann. 2007. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *ESWC*.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*.

R. Bunescu and R.Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*.

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

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to extract symbolic knowledge from the world wide web. In *AAAI*.

Dmitry Davidov and Ari Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *ACL*.

Dmitry Davidov, Ari Rappoport, and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *ACL*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. http://nlp.stanford.edu/downloads/lex-parser.shtml.

Benjamin Van Durme and Lenhart K. Schubert. 2008. Open knowledge extraction using compositional language processing. In *STEP*.

R. Hoffmann, C. Zhang, and D. Weld. 2010. Learning 5000 relational extractors. In *ACL*.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *HLT/NAACL*.

A. Gangemi M. Ciaramita. 2005. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *IJCAI*.

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*.

T. H. Kotaro Nakayama and S. Nishio. 2008. Wikipedia link structure and text mining for semantic relation extraction. In *CEUR Workshop*.

Dat P.T Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Exploiting syntactic and semantic information for relation extraction from wikipedia. In *IJCAI07-TextLinkWS*.

Marius Pasca. 2008. Turning web text and search queries into factual knowledge: Hierarchical class attribute extraction. In *AAAI*.

Fuchun Peng and Andrew McCallum. 2004. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *HLT-NAACL*.

Hoifung Poon and Pedro Domingos. 2008. Joint Inference in Information Extraction. In *AAAI*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unristricted relation discovery. In *HLT-NAACL*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In *WWW*.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *IJCNLP*.

Fei Wu and Daniel Weld. 2007. Autonomouslly Semantifying Wikipedia. In *CIKM*.

Fei Wu, Raphael Hoffmann, and Danel S. Weld. 2008. Information extraction from Wikipedia: Moving down the long tail. In *KDD*.

Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *ACL*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *ACL*.

Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*.

SystemT: An Algebraic Approach to Declarative Information Extraction

Laura Chiticariu Rajasekar Krishnamurthy Yunyao Li
Sriram Raghavan Frederick R. Reiss Shivakumar Vaithyanathan
IBM Research – Almaden
San Jose, CA, USA
{chiti,sekar,yunyaoli,rsriram,frreiss,vaithyan}@us.ibm.com

Abstract

As information extraction (IE) becomes more central to enterprise applications, rule-based IE engines have become increasingly important. In this paper, we describe SystemT, a rule-based IE system whose basic design removes the expressivity and performance limitations of current systems based on cascading grammars. SystemT uses a declarative rule language, AQL, and an optimizer that generates high-performance algebraic execution plans for AQL rules. We compare SystemT's approach against cascading grammars, both theoretically and with a thorough experimental evaluation. Our results show that SystemT can deliver result quality comparable to the state-of-the-art and an order of magnitude higher annotation throughput.

1 Introduction

In recent years, enterprises have seen the emergence of important text analytics applications like compliance and data redaction. This increase, combined with the inclusion of text into traditional applications like Business Intelligence, has dramatically increased the use of information extraction (IE) within the enterprise. While the traditional requirement of extraction quality remains critical, enterprise applications also demand efficiency, transparency, customizability and maintainability. In recent years, these systemic requirements have led to renewed interest in rule-based IE systems (Doan et al., 2008; SAP, 2010; IBM, 2010; SAS, 2010).

Until recently, rule-based IE systems (Cunningham et al., 2000; Boguraev, 2003; Drozdzynski et al., 2004) were predominantly based on the cascading grammar formalism exemplified by the Common Pattern Specification Language (CPSL) specification (Appelt and Onyshkevych, 1998). In CPSL, the input text is viewed as a sequence of annotations, and extraction rules are written as pattern/action rules over the lexical features of these annotations. In a single phase of the grammar, a set of rules are evaluated in a left-to-right fashion over the input annotations. Multiple grammar phases are cascaded together, with the evaluation proceeding in a bottom-up fashion.

As demonstrated by prior work (Grishman and Sundheim, 1996), grammar-based IE systems can be effective in many scenarios. However, these systems suffer from two severe drawbacks. First, the expressivity of CPSL falls short when used for complex IE tasks over increasingly pervasive informal text (emails, blogs, discussion forums etc.). To address this limitation, grammar-based IE systems resort to significant amounts of user-defined code in the rules, combined with pre- and post-processing stages beyond the scope of CPSL (Cunningham et al., 2010). Second, the rigid evaluation order imposed in these systems has significant performance implications.

Three decades ago, the database community faced similar expressivity and efficiency challenges in accessing structured information. The community addressed these problems by introducing a relational algebra formalism and an associated declarative query language SQL. The groundbreaking work on System R (Chamberlin et al., 1981) demonstrated how the expressivity of SQL can be efficiently realized in practice by means of a *query optimizer* that translates an SQL query into an optimized query execution plan.

Borrowing ideas from the database community, we have developed SystemT, a declarative IE system based on an algebraic framework, to address both expressivity and performance issues. In SystemT, extraction rules are expressed in a declarative language called AQL. At compilation time,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 128–137,
Uppsala, Sweden, 11-16 July 2010.

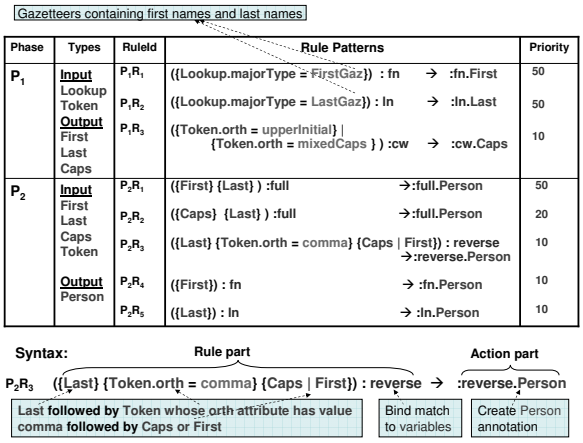


Figure 1: Cascading grammar for identifying Person names

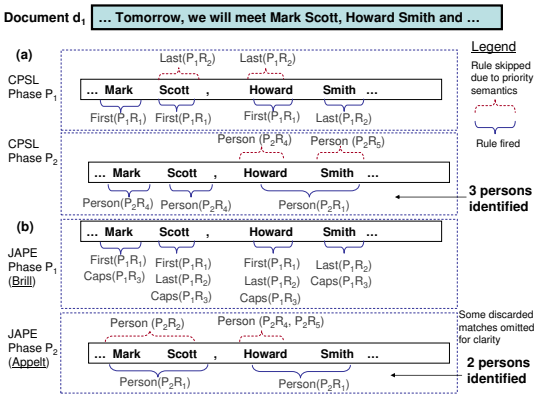


Figure 2: Sample output of CPSL and JAPE

SystemT translates AQL statements into an algebraic expression called an *operator graph* that implements the semantics of the statements. The SystemT optimizer then picks a fast execution plan from many logically equivalent plans. SystemT is currently deployed in a multitude of real-world applications and commercial products[1].

We formally demonstrate the superiority of AQL and SystemT in terms of both expressivity and efficiency (Section 4). Specifically, we show that 1) the expressivity of AQL is a strict superset of CPSL grammars not using external functions and 2) the search space explored by the SystemT optimizer includes operator graphs corresponding to efficient finite state transducer implementations. Finally, we present an extensive experimental evaluation that validates that high-quality annotators can be developed with SystemT, and that their runtime performance is an order of magnitude better when compared to annotators developed with a state-of-the-art grammar-based IE system (Section 5).

2 Grammar-based Systems and CPSL

A cascading grammar consists of a sequence of phases, each of which consists of one or more rules. Each phase applies its rules from left to right over an input sequence of annotations and generates an output sequence of annotations that the next phase consumes. Most cascading grammar systems today adhere to the CPSL standard.

Fig. 1 shows a sample CPSL grammar that identifies person names from text in two phases. The first phase, P_1, operates over the results of the tokenizer and gazetteer (input types *Token* and *Lookup*, respectively) to identify words that may be part of a person name. The second phase, P_2, identifies complete names using the results of phase P_1.

Applying the above grammar to document d_1 (Fig. 2), one would expect that to match "Mark Scott" and "Howard Smith" as *Person*. However, as shown in Fig. 2(a), the grammar actually finds three *Person* annotations, instead of two. CPSL has several limitations that lead to such discrepancies:

L1. Lossy sequencing. In a CPSL grammar, each phase operates on a sequence of annotations from left to right. If the input annotations to a phase may overlap with each other, the CPSL engine must drop some of them to create a non-overlapping sequence. For instance, in phase P_1 (Fig. 2(a)), "Scott" has both a *Lookup* and a *Token* annotation. The system has made an arbitrary choice to retain the *Lookup* annotation and discard the *Token* annotation. Consequently, no *Caps* annotations are output by phase P_1.

L2. Rigid matching priority. CPSL specifies that, for each input annotation, only one rule can actually match. When multiple rules match at the same start position, the following tie-breaker conditions are applied (in order): (a) the rule matching the most annotations in the input stream; (b) the rule with highest priority; and (c) the rule declared earlier in the grammar. This rigid matching priority can lead to mistakes. For instance, as illustrated in Fig. 2(a), phase P_1 only identifies "Scott" as a *First*. Matching priority causes the grammar to skip the corresponding match for "Scott" as a *Last*. Consequently, phase P_2 fails to identify "Mark Scott" as one single *Person*.

L3. Limited expressivity in rule patterns. It is not possible to express rules that compare annotations overlapping with each other. E.g., "Identify

[1] A trial version is available at *http://www.alphaworks.ibm.com/tech/systemt*

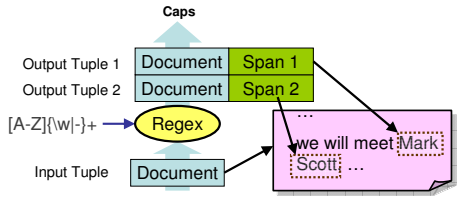


Figure 3: Regular Expression Extraction Operator

words that are both capitalized and present in the *FirstGaz* gazetteer" or "Identify *Person* annotations that occur within an *EmailAddress*".

Extensions to CPSL

In order to address the above limitations, several extensions to CPSL have been proposed in JAPE, AFst and XTDL (Cunningham et al., 2000; Boguraev, 2003; Drozdzynski et al., 2004). The extensions are summarized as below, where each solution S_i corresponds to limitation L_i.

- **S1**. Grammar rules are allowed to operate on graphs of input annotations in JAPE and AFst.
- **S2**. JAPE introduces more matching regimes besides the CPSL's matching priority and thus allows more flexibility when multiple rules match at the same starting position.
- **S3**. The rule part of a pattern has been expanded to allow more expressivity in JAPE, AFst and XTDL.

Fig. 2(b) illustrates how the above extensions help in identifying the correct matches '*Mark Scott*' and '*Howard Smith*' in JAPE. Phase P_1 uses a matching regime (denoted by *Brill*) that allows multiple rules to match at the same starting position, and phase P_2 uses CPSL's matching priority, *Appelt*.

3 SystemT

SystemT is a declarative IE system based on an algebraic framework. In SystemT, developers write rules in a language called AQL. The system then generates a graph of *operators* that implement the semantics of the AQL rules. This decoupling allows for greater rule expressivity, because the rule language is not constrained by the need to compile to a finite state transducer. Likewise, the decoupled approach leads to greater flexibility in choosing an efficient execution strategy, because many possible operator graphs may exist for the same AQL annotator.

In the rest of the section, we describe the parts of SystemT, starting with the algebraic formalism behind SystemT's operators.

3.1 Algebraic Foundation of SystemT

SystemT executes IE rules using graphs of operators. The formal definition of these operators takes the form of an algebra that is similar to the relational algebra, but with extensions for text processing.

The algebra operates over a simple relational data model with three data types: span, tuple, and relation. In this data model, a *span* is a region of text within a document identified by its "begin" and "end" positions; a *tuple* is a fixed-size list of spans. A *relation* is a multiset of tuples, where every tuple in the relation must be of the same size. Each *operator* in our algebra implements a single basic atomic IE operation, producing and consuming sets of tuples.

Fig. 3 illustrates the regular expression extraction operator in the algebra, which performs character-level regular expression matching. Overall, the algebra contains 12 different operators, a full description of which can be found in (Reiss et al., 2008). The following four operators are necessary to understand the examples in this paper:

- The **Extract** operator ($\mathcal{E}$) performs character-level operations such as regular expression and dictionary matching over text, creating a tuple for each match.
- The **Select** operator (σ) takes as input a set of tuples and a predicate to apply to the tuples. It outputs all tuples that satisfy the predicate.
- The **Join** operator ($\bowtie$) takes as input two sets of tuples and a predicate to apply to pairs of tuples from the input sets. It outputs all pairs of input tuples that satisfy the predicate.
- The **consolidate** operator (Ω) takes as input a set of tuples and the index of a particular column in those tuples. It removes selected overlapping spans from the indicated column, according to the specified policy.

3.2 AQL

Extraction rules in SystemT are written in AQL, a declarative relational language similar in syntax to the database language SQL. We chose SQL as a basis for our language due to its expressivity and its familiarity. The expressivity of SQL, which consists of first-order logic predicates

```
create view Caps as
extract regex /[A-Z](\w|-)+/ on D.text as name from Document D;

create view Last as
extract dictionary LastGaz on D.text as name from Document D;

create view CapsLast as
select  CombineSpans(C.name, L.name) as name
from    Caps C, Last L
where   FollowsTok(C.name, L.name, 0, 0);
...
create view PersonAll as
    (select R.name from FirstLast R) union all ...
                    ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;
```

Figure 4: *Person* annotator as AQL query

over sets of tuples, is well-documented and well-understood (Codd, 1990). As SQL is the primary interface to most relational database systems, the language's syntax and semantics are common knowledge among enterprise application programmers. Similar to SQL terminology, we call a collection of AQL rules an AQL *query*.

Fig. 4 shows portions of an AQL query. As can be seen, the basic building block of AQL is a *view*: A logical description of a set of tuples in terms of either the document text (denoted by a special view called `Document`) or the contents of other views. Every SystemT annotator consists of at least one view. The *output view* statement indicates that the tuples in a view are part of the final results of the annotator.

Fig. 4 also illustrates three of the basic constructs that can be used to define a view.

- The **`extract`** statement specifies basic character-level extraction primitives to be applied directly to a tuple.
- The **`select`** statement is similar to the SQL `select` statement but it contains an additional `consolidate on` clause, along with an extensive collection of text-specific predicates.
- The **`union all`** statement merges the outputs of one or more `select` or `extract` statements.

To keep rules compact, AQL also provides a shorthand *sequence pattern* notation similar to the syntax of CPSL. For example, the `CapsLast` view in Figure 4 could have been written as:

```
create view CapsLast as
extract pattern <C.name> <L.name>
from Caps C, Last L;
```

Internally, SystemT translates each of these *extract pattern* statements into one or more *select* and *extract* statements.

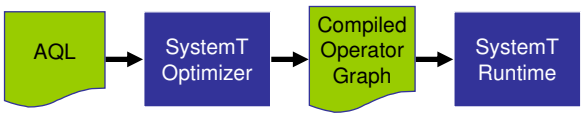


Figure 5: The compilation process in SystemT

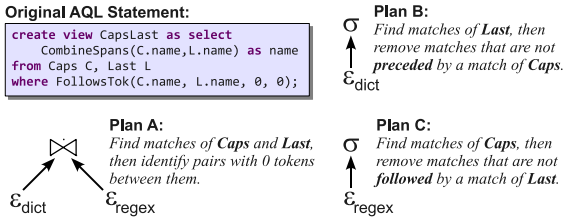


Figure 6: Execution strategies for the *CapsLast* rule in Fig. 4

SystemT has built-in multilingual support including tokenization, part of speech and gazetteer matching for over 20 languages using LanguageWare (IBM, 2010). Rule developers can utilize the multilingual support via AQL without having to configure or manage any additional resources. In addition, AQL allows user-defined functions to be used in a restricted context in order to support operations such as validation (e.g. for extracted credit card numbers), or normalization (e.g., compute abbreviations of multi-token organization candidates that are useful in generating additional candidates). More details on AQL can be found in the AQL manual (SystemT, 2010).

3.3 Optimizer and Operator Graph

Grammar-based IE engines place rigid restrictions on the order in which rules can be executed. Due to the semantics of the CPSL standard, systems that implement the standard must use a finite state transducer that evaluates each level of the cascade with one or more left to right passes over the entire token stream.

In contrast, SystemT places no explicit constraints on the order of rule evaluation, nor does it require that intermediate results of an annotator collapse to a fixed-size sequence. As shown in Fig. 5, the SystemT engine does not execute AQL directly; instead, the SystemT *optimizer* compiles AQL into a graph of operators. By tying a collection of operators together by their inputs and outputs, the system can implement a wide variety of different execution strategies. Different execution strategies are associated with different evaluation costs. The optimizer chooses the execution strategy with the lowest estimated evaluation cost.

Fig. 6 presents three possible execution strategies for the *CapsLast* rule in Fig. 4. If the optimizer estimates that the evaluation cost of *Last* is much lower than that of *Caps*, then it can determine that Plan C has the lowest evaluation cost among the three, because Plan C only evaluates *Caps* in the "left" neighborhood for each instance of *Last*. More details of our algorithms for enumerating plans can be found in (Reiss et al., 2008).

The optimizer in SystemT chooses the best execution plan from a large number of different algebra graphs available to it. Many of these graphs implement strategies that a transducer could not express: such as evaluating rules from right to left, sharing work across different rules, or selectively skipping rule evaluations. Within this large search space, there generally exists an execution strategy that implements the rule semantics far more efficiently than the fastest transducer could. We refer the reader to (Reiss et al., 2008) for a detailed description of the types of plan the optimizer considers, as well as an experimental analysis of the performance benefits of different parts of this search space.

Several parallel efforts have been made recently to improve the efficiency of IE tasks by optimizing low-level feature extraction (Ramakrishnan et al., 2006; Ramakrishnan et al., 2008; Chandel et al., 2006) or by reordering operations at a macroscopic level (Ipeirotis et al., 2006; Shen et al., 2007; Jain et al., 2009). However, to the best of our knowledge, SystemT is the only IE system in which the optimizer generates a full end-to-end plan, beginning with low-level extraction primitives and ending with the final output tuples.

3.4 Deployment Scenarios

SystemT is designed to be usable in various deployment scenarios. It can be used as a standalone system with its own development and runtime environment. Furthermore, SystemT exposes a generic Java API that enables the integration of its runtime environment with other applications. For example, a specific instantiation of this API allows SystemT annotators to be seamlessly embedded in applications using the UIMA analytics framework (UIMA, 2010).

4 Grammar vs. Algebra

Having described both the traditional cascading grammar approach and the declarative approach

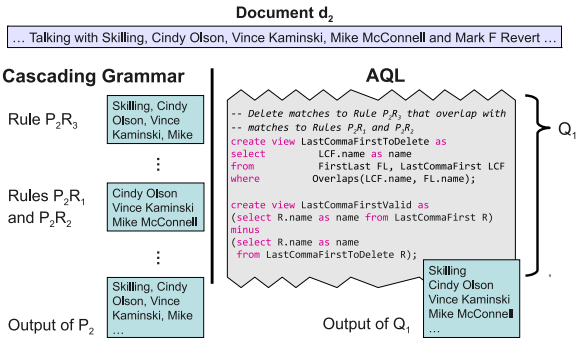


Figure 7: Supporting Complex Rule Interactions

used in SystemT, we now compare the two in terms of expressivity and performance.

4.1 Expressivity

In Section 2, we described three expressivity limitations of CPSL grammars: Lossy sequencing, rigid matching priority, and limited expressivity in rule patterns. As we noted, cascading grammar systems extend the CPSL specification in various ways to provide workarounds for these limitations.

In SystemT, the basic design of the AQL language eliminates these three problems without the need for any special workaround. The key design difference is that AQL views operate over sets of tuples, not sequences of tokens. The input or output tuples of a view can contain spans that overlap in arbitrary ways, so the lossy sequencing problem never occurs. The annotator will retain these overlapping spans across any number of views until a view definition explicitly removes the overlap. Likewise, the tuples that a given view produces are in no way constrained by the outputs of other, unrelated views, so the rigid matching priority problem never occurs. Finally, the *select* statement in AQL allows arbitrary predicates over the cross-product of its input tuple sets, eliminating the limited expressivity in rule patterns problem.

Beyond eliminating the major limitations of CPSL grammars, AQL provides a number of other information extraction operations that even extended CPSL cannot express without custom code.

Complex rule interactions. Consider an example document from the Enron corpus (Minkov et al., 2005), shown in Fig. 7, which contains a list of person names. Because the first person in the list ('*Skilling*') is referred to by only a last name, rule P_2R_3 in Fig. 1 incorrectly identifies '*Skilling, Cindy*' as a person. Consequently, the output of phase P_2 of the cascading grammar contains several mistakes as shown in the figure. This problem

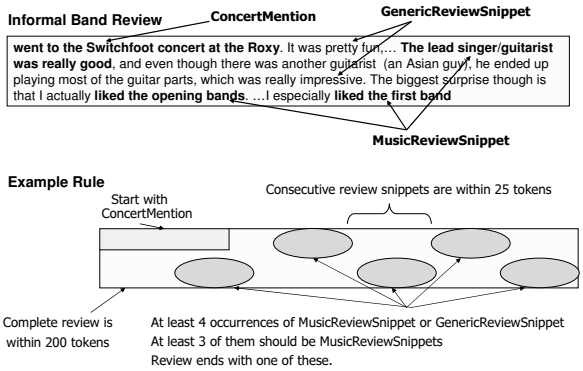


Figure 8: Extracting informal band reviews from web logs

occurs because CPSL only evaluates rules over the input sequence in a strict left-to-right fashion. On the other hand, the AQL query Q_1 shown in the figure applies the following condition: "Always discard matches to Rule P_2R_3 if they overlap with matches to rules P_2R_1 or P_2R_2" (even if the match to Rule P_2R_3 starts earlier). Applying this rule ensures that the person names in the list are identified correctly. Obtaining the same effect in grammar-based systems would require the use of custom code (as recommended by (Cunningham et al., 2010)).

Counting and Aggregation. Complex extraction tasks sometimes require operations such as counting and aggregation that go beyond the expressivity of regular languages, and thus can be expressed in CPSL only using external functions. One such task is that of identifying informal concert reviews embedded within blog entries. Fig. 8 describes, by example, how these reviews consist of reference to a live concert followed by several review snippets, some specific to musical performances and others that are more general review expressions. An example rule to identify informal reviews is also shown in the figure. Notice how implementing this rule requires *c*ounting the number of *MusicReviewSnippet* and *GenericReviewSnippet* annotations within a region of text and *a*ggregating this occurrence count across the two review types. While this rule can be written in AQL, it can only be approximated in CPSL grammars.

Character-Level Regular Expression CPSL cannot specify character-level regular expressions that span multiple tokens. In contrast, the *extract regex* statement in AQL fully supports these expressions.

We have described above several cases where AQL can express concepts that can only be expressed through external functions in a cascading grammar. These examples naturally raise the question of whether similar cases exist where a cascading grammar can express patterns that cannot be expressed in AQL.

It turns out that we can make a strong statement that such examples do not exist. In the absence of an escape to arbitrary procedural code, AQL is strictly more expressive than a CPSL grammar. To state this relationship formally, we first introduce the following definitions.

We refer to a grammar conforming to the CPSL specification as a *CPSL grammar*. When a CPSL grammar contains no external functions, we refer to it as a *Code-free CPSL grammar*. Finally, we refer to a grammar that conforms to one of the CPSL, JAPE, AFst and XTDL specifications as an *expanded CPSL grammar*.

Ambiguous Grammar Specification An *expanded CPSL grammar* may be under-specified in some cases. For example, a single rule containing the disjunction operator (|) may match a given region of text in multiple ways. Consider the evaluation of Rule P_2R_3 over the text fragment "Scott, Howard" from document d_1 (Fig. 1). If "Howard" is identified both as *Caps* and *First*, then there are two evaluations for Rule P_2R_3 over this text fragment. Since the system has to arbitrarily choose one evaluation, the results of the grammar can be non-deterministic (as pointed out in (Cunningham et al., 2010)). We refer to a grammar G as an *ambiguous grammar specification* for a document collection $\mathcal{D}$ if the system makes an arbitrary choice while evaluating G over $\mathcal{D}$.

Definition 1 (*UnambigEquiv*) *A query Q is UnambigEquiv to a cascading grammar $\mathcal{G}$ if and only if for every document collection $\mathcal{D}$, where G is not an ambiguous grammar specification for $\mathcal{D}$, the results of the grammar invocation and the query evaluation are identical.*

We now formally compare the expressivity of AQL and expanded CPSL grammars. The detailed proof is omitted due to space limitations.

Theorem 1 *The class of extraction tasks expressible as AQL queries is a strict superset of that expressible through expanded code-free CPSL grammars. Specifically,*
(a) Every expanded code-free CPSL grammar can be expressed as an UnambigEquiv AQL query.
(b) AQL supports information extraction operations that cannot be expressed in expanded code-free CPSL grammars.

Proof Outline: *(a)* A single CPSL grammar can be expressed in AQL as follows. First, each rule r in the grammar is translated into a set of AQL statements. If r does not contain the disjunct (|) operator, then it is translated into a single AQL *select* statement. Otherwise, a set of AQL statements are generated, one for each disjunct operator in rule r, and the results merged using *union all* statements. Then, a *union all* statement is used to combine the results of individual rules in the grammar phase. Finally, the AQL statements for multiple phases are combined in the same order as the cascading grammar specification.

The main extensions to CPSL supported by expanded CPSL grammars (listed in Sec. 2) are handled as follows. AQL queries operate on graphs on annotations just like expanded CPSL grammars. In addition, AQL supports different matching regimes through consolidation operators, span predicates through selection predicates and coreferences through join operators.
(b) Example operations supported in AQL that cannot be expressed in expanded code-free CPSL grammars include (i) character-level regular expressions spanning multiple tokens, (ii) counting the number of annotations occurring within a given bounded window and (iii) deleting annotations if they overlap with other annotations starting later in the document. □

4.2 Performance

For the annotators we test in our experiments (See Section 5), the SystemT optimizer is able to choose algebraic plans that are faster than a comparable transducer-based implementation. The question arises as to whether there are other annotators for which the traditional transducer approach is superior. That is, for a given annotator, might there exist a finite state transducer that is combinatorially faster than any possible algebra graph? It turns out that this scenario is not possible, as the theorem below shows.

Definition 2 (Token-Based FST) *A* token-based finite state transducer *(FST) is a nondeterministic finite state machine in which state transitions are triggered by predicates on tokens. A token-based FST is* acyclic *if its state graph does not contain any cycles and has exactly one "accept" state.*

Definition 3 (Thompson's Algorithm) Thompson's algorithm *is a common strategy for evaluating a token-based FST (based on (Thompson, 1968)). This algorithm processes the input tokens from left to right, keeping track of the set of states that are currently active.*

Theorem 2 *For any acyclic token-based finite state transducer T, there exists an UnambigEquiv operator graph G, such that evaluating G has the same computational complexity as evaluating T with Thompson's algorithm starting from each token position in the input document.*

Proof Outline: The proof constructs G by structural induction over the transducer T. The base case converts transitions out of the start state into *Extract* operators. The inductive case adds a *Select* operator to G for each of the remaining state transitions, with each selection predicate being the same as the predicate that drives the corresponding state transition. For each state transition predicate that T would evaluate when processing a given document, G performs a constant amount of work on a single tuple. □

5 Experimental Evaluation

In this section we present an extensive comparison study between SystemT and implementations of expanded CPSL grammar in terms of quality, runtime performance and resource requirements.
Tasks We chose two tasks for our evaluation:

- NER : named-entity recognition for Person, Organization, Location, Address, PhoneNumber, EmailAddress, URL and DateTime.
- BandReview : identify informal reviews in blogs (Fig. 8).

We chose NER primarily because named-entity recognition is a well-studied problem and standard datasets are available for evaluation. For this task we use GATE and ANNIE for comparison[3]. We chose BandReview to conduct performance evaluation for a more complex extraction task.
Datasets. For quality evaluation, we use:

- *EnronMeetings* (Minkov et al., 2005): collection of emails with meeting information from the Enron corpus[4] with Person labeled data;
- *ACE* (NIST, 2005): collection of newswire reports and broadcast news/conversations with Person, Organization, Location labeled data[5].

[3]To the best of our knowledge, ANNIE (Cunningham et al., 2002) is the only publicly available NER library implemented in a grammar-based system (JAPE in GATE).
[4]http://www.cs.cmu.edu/ enron/
[5]Only entities of type NAM have been considered.

Table 1: Datasets for performance evaluation.

Dataset	*Description of the Content*	*Number of documents*	*Document size*	
			range	*average*
$Enron_x$	Emails randomly sampled from the Enron corpus of average size xKB $(0.5 < x < 100)$[2]	1000	xKB $+/-$ 10%	xKB
WebCrawl	Small to medium size web pages representing company news, with HTML tags removed	1931	68b - 388.6KB	8.8KB
$Finance_M$	Medium size financial regulatory filings	100	240KB - 0.9MB	401KB
$Finance_L$	Large size financial regulatory filings	30	1MB - 3.4MB	1.54MB

Table 2: Quality of Person on test datasets.

	Precision (%) (Exact/Partial)	*Recall (%) (Exact/Partial)*	*F1 measure (%) (Exact/Partial)*
EnronMeetings			
ANNIE	57.05/76.84	48.59/65.46	52.48/70.69
T-NE	**88.41/92.99**	**82.39/86.65**	**85.29/89.71**
Minkov	81.1/NA	74.9/NA	77.9/NA
ACE			
ANNIE	39.41/78.15	30.39/60.27	34.32/68.06
T-NE	**93.90/95.82**	**90.90/92.76**	**92.38/94.27**

Table 1 lists the datasets used for performance evaluation. The size of $Finance_L$ is purposely small because GATE takes a significant amount of time processing large documents (see Sec. 5.2).

Set Up. The experiments were run on a server with two 2.4 GHz 4-core Intel Xeon CPUs and 64GB of memory. We use GATE 5.1 (build 3431) and two configurations for ANNIE: 1) the *default* configuration, and 2) an *optimized* configuration where the Ontotext Japec Transducer[6] replaces the default NE transducer for optimized performance. We refer to these configurations as ANNIE and ANNIE-Optimized, respectively.

5.1 Quality Evaluation

The goal of our quality evaluation is two-fold: to validate that annotators can be built in SystemT with quality comparable to those built in a grammar-based system; and to ensure a fair performance comparison between SystemT and GATE by verifying that the annotators used in the study are comparable.

Table 2 shows results of our comparison study for Person annotators. We report the classical (*exact*) precision, recall, and $F1$ measures that credit only exact matches, and corresponding *partial* measures that credit partial matches in a fashion similar to (NIST, 2005). As can be seen, T-NE produced results of significantly higher quality than ANNIE on both datasets, for the same Person extraction task. In fact, on *EnronMeetings*, the $F1$ measure of T-NE is 7.4% higher than the best published result (Minkov et al., 2005). Similar results

[6] http://www.ontotext.com/gate/japec.html

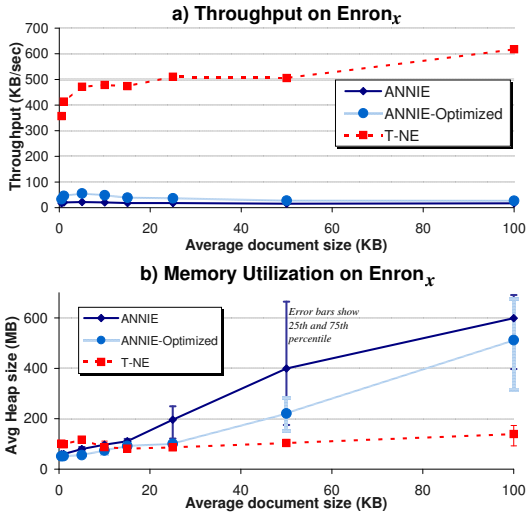


Figure 9: Throughput (a) and memory consumption (b) comparisons on $Enron_x$ datasets.

can be observed for Organization and Location on *ACE* (exact numbers omitted in interest of space).

Clearly, considering the large gap between ANNIE's $F1$ and partial $F1$ measures on both datasets, ANNIE's quality can be improved via dataset-specific tuning as demonstrated in (Maynard et al., 2003). However, dataset-specific tuning for ANNIE is beyond the scope of this paper. Based on the experimental results above and our previous formal comparison in Sec. 4, we believe it is reasonable to conclude that annotators can be built in SystemT of quality at least comparable to those built in a grammar-based system.

5.2 Performance Evaluation

We now focus our attention on the throughput and memory behavior of SystemT, and draw a comparison with GATE. For this purpose, we have configured both ANNIE and T-NE to identify only the same eight types of entities listed for NER task.

Throughput. Fig. 9(a) plots the throughput of the two systems on multiple $Enron_x$ datasets with average document sizes of between 0.5KB and 100KB. For this experiment, both systems ran with a maximum Java heap size of 1GB.

Table 3: Throughput and mean heap size.

Dataset	ANNIE		ANNIE-Optimized		T-NE	
	Throughput (KB/s)	*Memory (MB)*	*Throughput (KB/s)*	*Memory (MB)*	*Throughput (KB/s)*	*Memory (MB)*
WebCrawl	23.9	212.6	42.8	201.8	**498.9**	**77.2**
$Finance_M$	18.82	715.1	26.3	601.8	**703.5**	**143.7**
$Finance_L$	19.2	2586.2	21.1	2683.5	**954.5**	**189.6**

As shown in Fig. 9(a), even though the throughput of ANNIE-Optimized (using the optimized transducer) increases two-fold compared to ANNIE under default configuration, T-NE is between 8 and 24 times faster compared to ANNIE-Optimized. For both systems, throughput varied with document size. For T-NE, the relatively low throughput on very small document sizes (less than 1KB) is due to fixed overhead in setting up operators to process a document. As document size increases, the overhead becomes less noticeable.

We have observed similar trends on the rest of the test collections. Table 3 shows that T-NE is at least an order of magnitude faster than ANNIE-Optimized across all datasets. In particular, on $Finance_L$ T-NE's throughput remains high, whereas the performance of both ANNIE and ANNIE-Optimized degraded significantly.

To ascertain whether the difference in performance in the two systems is due to low-level components such as dictionary evaluation, we performed detailed profiling of the systems. The profiling revealed that 8.2%, 16.2% and respectively 14.2% of the execution time was spent on average on low-level components in the case of ANNIE, ANNIE-Optimized and T-NE, respectively, thus leading us to conclude that the observed differences are due to SystemT's efficient use of resources at a macroscopic level.

Memory utilization. In theory, grammar based systems can stream tuples through each stage for minimal memory consumption, whereas SystemT operator graphs may need to materialize intermediate results for the full document at certain points to evaluate the constraints in the original AQL. The goal of this study is to evaluate whether this potential problem does occur in practice.

In this experiment we ran both systems with a maximum heap size of 2GB, and used the Java garbage collector's built-in telemetry to measure the total quantity of live objects in the heap over time while annotating the different test corpora. Fig. 9(b) plots the minimum, maximum, and mean heap sizes with the $Enron_x$ datasets. On small documents of size up to 15KB, memory consumption is dominated by the fixed size of the data structures used (e.g., dictionaries, FST/operator graph), and is comparable for both systems. As documents get larger, memory consumption increases for both systems. However, the increase is much smaller for T-NE compared to that for both ANNIE and ANNIE-Optimized. A similar trend can be observed on the other datasets as shown in Table 3. In particular, for $Finance_L$, both ANNIE and ANNIE-Optimized required 8GB of Java heap size to achieve reasonable throughput[7], in contrast to T-NE which utilized at most 300MB out of the 2GB of maximum Java heap size allocation.

SystemT requires much less memory than GATE in general due to its runtime, which monitors data dependencies between operators and clears out low-level results when they are no longer needed. Although a streaming CPSL implementation is theoretically possible, in practice mechanisms that allow an escape to custom code make it difficult to decide when an intermediate result will no longer be used, hence GATE keeps most intermediate data in memory until it is done analyzing the current document.

The BandReview Task. We conclude by briefly discussing our experience with the BandReview task from Fig. 8. We built two versions of this annotator, one in AQL, and the other using expanded CPSL grammar. The grammar implementation processed a 4.5GB collection of 1.05 million blogs in 5.6 hours and output 280 reviews. In contrast, the SystemT version (85 AQL statements) extracted 323 reviews in only 10 minutes!

6 Conclusion

In this paper, we described SystemT, a declarative IE system based on an algebraic framework. We presented both formal and empirical arguments for the benefits of our approach to IE. Our extensive experimental results show that high-quality annotators can be built using SystemT, with an order of magnitude throughput improvement compared to state-of-the-art grammar-based systems. Going forward, SystemT opens up several new areas of research, including implementing better optimization strategies and augmenting the algebra with additional operators to support advanced features such as coreference resolution.

[7] GATE ran out of memory when using less than 5GB of Java heap size, and thrashed when run with 5GB to 7GB

References

Douglas E. Appelt and Boyan Onyshkevych. 1998. The common pattern specification language. In *TIPSTER workshop*.

Branimir Boguraev. 2003. Annotation-based finite state processing in a large-scale nlp arhitecture. In *RANLP*, pages 61–80.

D. D. Chamberlin, A. M. Gilbert, and Robert A. Yost. 1981. A history of System R and SQL/data system. In *vldb*.

Amit Chandel, P. C. Nagesh, and Sunita Sarawagi. 2006. Efficient batch top-k search for dictionary-based entity recognition. In *ICDE*.

E. F. Codd. 1990. *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, pages 168 – 175.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Marin Dimitrov, Mike Dowman, Niraj Aswani, Ian Roberts, Yaoyong Li, and Adam Funk. 2010. Developing language processing components with gate version 5 (a user guide).

AnHai Doan, Luis Gravano, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. 2008. Special issue on managing information extraction. *SIGMOD Record*, 37(4).

Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference - 6: A brief history. In *COLING*, pages 466–471.

IBM. 2010. IBM LanguageWare.

P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano. 2006. To search or to crawl?: towards a query optimizer for text-centric tasks. In *SIGMOD*.

Alpa Jain, Panagiotis G. Ipeirotis, AnHai Doan, and Luis Gravano. 2009. Join optimization of information extraction output: Quality matters! In *ICDE*.

Diana Maynard, Kalina Bontcheva, and Hamish Cunningham. 2003. Towards a semantic extraction of named entities. In *Recent Advances in Natural Language Processing*.

Einat Minkov, Richard C. Wang, and William W. Cohen. 2005. Extracting personal names from emails: Applying named entity recognition to informal text. In *HLT/EMNLP*.

NIST. 2005. The ACE evaluation plan.

Ganesh Ramakrishnan, Sreeram Balakrishnan, and Sachindra Joshi. 2006. Entity annotation based on inverse index operations. In *EMNLP*.

Ganesh Ramakrishnan, Sachindra Joshi, Sanjeet Khaitan, and Sreeram Balakrishnan. 2008. Optimization issues in inverted index-based entity annotation. In *InfoScale*.

Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. 2008. An algebraic approach to rule-based information extraction. In *ICDE*, pages 933–942.

SAP. 2010. Inxight ThingFinder.

SAS. 2010. Text Mining with SAS Text Miner.

Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. 2007. Declarative information extraction using datalog with embedded extraction predicates. In *vldb*.

SystemT. 2010. AQL Manual. http://www.alphaworks.ibm.com/tech/systemt.

Ken Thompson. 1968. Regular expression search algorithm. pages 419–422.

UIMA. 2010. Unstructured Information Management Architecture. http://uima.apache.org.

Extracting Social Networks from Literary Fiction

David K. Elson
Dept. of Computer Science
Columbia University
delson@cs.columbia.edu

Nicholas Dames
English Department
Columbia University
nd122@columbia.edu

Kathleen R. McKeown
Dept. of Computer Science
Columbia University
kathy@cs.columbia.edu

Abstract

We present a method for extracting social networks from literature, namely, nineteenth-century British novels and serials. We derive the networks from dialogue interactions, and thus our method depends on the ability to determine when two characters are in conversation. Our approach involves character name chunking, quoted speech attribution and conversation detection given the set of quotes. We extract features from the social networks and examine their correlation with one another, as well as with metadata such as the novel's setting. Our results provide evidence that the majority of novels in this time period do not fit two characterizations provided by literacy scholars. Instead, our results suggest an alternative explanation for differences in social networks.

1 Introduction

Literary studies about the nineteenth-century British novel are often concerned with the nature of the community that surrounds the protagonist. Some theorists have suggested a relationship between the size of a community and the amount of dialogue that occurs, positing that "face to face time" diminishes as the number of characters in the novel grows. Others suggest that as the social setting becomes more urbanized, the quality of dialogue also changes, with more interactions occurring in rural communities than urban communities. Such claims have typically been made, however, on the basis of a few novels that are studied in depth. In this paper, we aim to determine whether an automated study of a much larger sample of nineteenth century novels supports these claims.

The research presented here is concerned with the extraction of social networks from literature. We present a method to automatically construct a network based on dialogue interactions between characters in a novel. Our approach includes components for finding instances of quoted speech, attributing each quote to a character, and identifying when certain characters are in conversation. We then construct a network where characters are vertices and edges signify an amount of bilateral conversation between those characters, with edge weights corresponding to the frequency and length of their exchanges. In contrast to previous approaches to social network construction, ours relies on a novel combination of pattern-based detection, statistical methods, and adaptation of standard natural language tools for the literary genre. We carried out this work on a corpus of 60 nineteenth-century novels and serials, including 31 authors such as Dickens, Austen and Conan Doyle.

In order to evaluate the literary claims in question, we compute various characteristics of the dialogue-based social network and stratify these results by categories such as the novel's setting. For example, the density of the network provides evidence about the cohesion of a large or small community, and cliques may indicate a social fragmentation. Our results surprisingly provide evidence that the majority of novels in this time period do not fit the suggestions provided by literary scholars, and we suggest an alternative explanation for our observations of differences across novels.

In the following sections, we survey related work on social networks as well as computational studies of literature. We then present the literary hypotheses in more detail. We describe the methods we use to extract dialogue and construct conversational networks, along with our approach to analyzing their characteristics. After we present the statistical results, we analyze their significance from a literary perspective.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 138–147,
Uppsala, Sweden, 11-16 July 2010.

2 Related Work

Computer-assisted literary analysis has typically occurred at the word level. This level of granularity lends itself to studies of authorial style based on patterns of word use (Burrows, 2004), and researchers have successfully "outed" the writers of anonymous texts by comparing their style to that of a corpus of known authors (Mostellar and Wallace, 1984). Determining instances of "text reuse," a type of paraphrasing, is also a form of analysis at the lexical level, and it has recently been used to validate theories about the lineage of ancient texts (Lee, 2007).

Analysis of literature using more semantically-oriented techniques has been rare, most likely because of the difficulty in automatically determining meaningful interpretations. Some exceptions include recent work on learning common event sequences in news stories (Chambers and Jurafsky, 2008), an approach based on statistical methods, and the development of an event calculus for characterizing stories written by children (Halpin et al., 2004), a knowledge-based strategy. On the other hand, literary theorists, linguists and others have long developed symbolic but non-computational models for novels. For example, Moretti (2005) has graphically mapped out texts according to geography, social connections and other variables.

While researchers have not attempted the automatic construction of social networks representing connections between characters in a corpus of novels, the ACE program has involved entity and relation extraction in unstructured text (Doddington et al., 2004). Other recent work in social network construction has explored the use of structured data such as email headers (McCallum et al., 2007) and U.S. Senate bill cosponsorship (Cho and Fowler, 2010). In an analysis of discussion forums, Gruzd and Haythornthwaite (2008) explored the use of message text as well as posting data to infer who is talking to whom. In this paper, we also explore how to build a network based on conversational interaction, but we analyze the reported dialogue found in novels to determine the links. The kinds of language that is used to signal such information is quite different in the two media. In discussion forums, people tend to use addresses such as "Hi Tom," while in novels, a system must determine both the speaker of a quotation and then the intended recipient of the dialogue act. This is a significantly different problem.

3 Hypotheses

It is commonly held that the novel is a literary form which tries to produce an accurate representation of the social world. Within literary studies, the recurring problem is how that representation is achieved. Theories about the relation between novelistic form (the workings of plot, characters, and dialogue, to take the most basic categories) and changes to real-world social milieux abound. Many of these theories center on nineteenth-century European fiction; innovations in novelistic form during this period, as well as the rapid social changes brought about by revolution, industrialization, and transport development, have traditionally been linked. These theories, however, have used only a select few representative novels as proof. By using statistical methods of analysis, it is possible to move beyond this small corpus of proof texts. We believe these methods are essential to testing the validity of some core theories about social interaction and their representation in literary genres like the novel.

Major versions of the theories about the social worlds of nineteenth-century fiction tend to center on characters, in two specific ways: how many characters novels tend to have, and how those characters interact with one another. These two "formal" facts about novels are usually explained with reference to a novel's setting. From the influential work of the Russian critic Mikhail Bakhtin to the present, a consensus emerged that as novels are increasingly set in urban areas, the number of characters and the quality of their interaction change to suit the setting. Bakhtin's term for this causal relationship was *chronotope*: the "intrinsic interconnectedness of temporal and spatial relationships that are artistically expressed in literature," in which "space becomes charged and responsive to movements of time, plot, and history" (Bakhtin, 1981, 84). In Bakhtin's analysis, different spaces have different social and emotional potentialities, which in turn affect the most basic aspects of a novel's aesthetic technique.

After Bakhtin's invention of the chronotope, much literary criticism and theory devoted itself to filling in, or describing, the qualities of specific chronotopes, particularly those of the village or rural environment and the city or urban environment. Following a suggestion of Bakhtin's that the population of village or rural fictions is modeled on the world of the family, made up of

Author/Title/Year	Persp.	Setting	Author/Title/Year	Persp.	Setting
Ainsworth, *Jack Sheppard* (1839)	3rd	urban	Gaskell, *North and South* (1854)	3rd	urban
Austen, *Emma* (1815)	3rd	rural	Gissing, *In the Year of Jubilee* (1894)	3rd	urban
Austen, *Mansfield Park* (1814)	3rd	rural	Gissing, *New Grub Street* (1891)	3rd	urban
Austen, *Persuasion* (1817)	3rd	rural	Hardy, *Jude the Obscure* (1894)	3rd	mixed
Austen, *Pride and Prejudice* (1813)	3rd	rural	Hardy, *The Return of the Native* (1878)	3rd	rural
Braddon, *Lady Audley's Secret* (1862)	3rd	mixed	Hardy, *Tess of the d'Ubervilles* (1891)	3rd	rural
Braddon, *Aurora Floyd* (1863)	3rd	rural	Hughes, *Tom Brown's School Days* (1857)	3rd	rural
Brontë, Anne, *The Tenant of Wildfell Hall* (1848)	1st	rural	James, *The Portrait of a Lady* (1881)	3rd	urban
Brontë, Charlotte, *Jane Eyre* (1847)	1st	rural	James, *The Ambassadors* (1903)	3rd	urban
Brontë, Charlotte, *Villette* (1853)	1st	mixed	James, *The Wings of the Dove* (1902)	3rd	urban
Brontë, Emily, *Wuthering Heights* (1847)	1st	rural	Kingsley, *Alton Locke* (1860)	1st	mixed
Bulwer-Lytton, *Paul Clifford* (1830)	3rd	urban	Martineau, *Deerbrook* (1839)	3rd	rural
Collins, *The Moonstone* (1868)	1st	urban	Meredith, *The Egoist* (1879)	3rd	rural
Collins, *The Woman in White* (1859)	1st	urban	Meredith, *The Ordeal of Richard Feverel* (1859)	3rd	rural
Conan Doyle, *The Sign of the Four* (1890)	1st	urban	Mitford, *Our Village* (1824)	1st	rural
Conan Doyle, *A Study in Scarlet* (1887)	1st	urban	Reade, *Hard Cash* (1863)	3rd	urban
Dickens, *Bleak House* (1852)	mixed	urban	Scott, *The Bride of Lammermoor* (1819)	3rd	rural
Dickens, *David Copperfield* (1849)	1st	mixed	Scott, *The Heart of Mid-Lothian* (1818)	3rd	rural
Dickens, *Little Dorrit* (1855)	3rd	urban	Scott, *Waverley* (1814)	3rd	rural
Dickens, *Oliver Twist* (1837)	3rd	urban	Stevenson, *The Strange Case of Dr. Jekyll and Mr. Hyde* (1886)	1st	urban
Dickens, *The Pickwick Papers* (1836)	3rd	mixed	Stoker, *Dracula* (1897)	1st	urban
Disraeli, *Sybil, or the Two Nations* (1845)	3rd	mixed	Thackeray, *History of Henry Esmond* (1852)	1st	urban
Edgeworth, *Belinda* (1801)	3rd	rural	Thackeray, *History of Pendennis* (1848)	1st	urban
Edgeworth, *Castle Rackrent* (1800)	3rd	rural	Thackeray, *Vanity Fair* (1847)	3rd	urban
Eliot, *Adam Bede* (1859)	3rd	rural	Trollope, *Barchester Towers* (1857)	3rd	rural
Eliot, *Daniel Deronda* (1876)	3rd	urban	Trollope, *Doctor Thorne* (1858)	3rd	rural
Eliot, *Middlemarch* (1871)	3rd	rural	Trollope, *Phineas Finn* (1867)	3rd	urban
Eliot, *The Mill on the Floss* (1860)	3rd	rural	Trollope, *The Way We Live Now* (1874)	3rd	urban
Galt, *Annals of the Parish* (1821)	1st	rural	Wilde, *The Picture of Dorian Gray* (1890)	3rd	urban
Gaskell, *Mary Barton* (1848)	3rd	urban	Wood, *East Lynne* (1860)	3rd	mixed

Table 1: Properties of the nineteenth-century British novels and serials included in our study.

an intimately related set of characters, many critics analyzed the formal expression of this world as constituted by a small set of characters who express themselves conversationally. Raymond Williams used the term "knowable communities" to describe this world, in which face-to-face relations of a restricted set of characters are the primary mode of social interaction (Williams, 1975, 166).

By contrast, the urban world, in this traditional account, is both larger and more complex. To describe the social-psychological impact of the city, Franco Moretti argues, protagonists of urban novels "change overnight from 'sons' into 'young men': their affective ties are no longer vertical ones (between successive generations), but horizontal, within the same generation. They are drawn towards those unknown yet congenial faces seen in gardens, or at the theater; future friends, or rivals, or both" (Moretti, 1999, 65). The result is two-fold: more characters, indeed a mass of characters, and more interactions, although less actual conversation; as literary critic Terry Eagleton argues, the city is where "most of our encounters consist of seeing rather than speaking, glimpsing each other as objects rather than conversing as fellow subjects" (Eagleton, 2005, 145). Moretti argues in similar terms. For him, the difference in number of characters is "not just a matter of quantity... it's a qualitative, morphological one" (Moretti, 1999, 68). As the number of characters increases, Moretti argues (following Bakhtin in his logic), social interactions of different kinds and durations multiply, displacing the family-centered and conversational logic of village or rural fictions. "The narrative system becomes complicated, unstable: the city turns into a gigantic roulette table, where helpers and antagonists mix in unpredictable combinations" (Moretti, 1999, 68). This argument about how novelistic setting produces different forms of social interaction is precisely what our method seeks to evaluate.

Our corpus of 60 novels was selected for its representativeness, particularly in the following categories: authorial (novels from the major canoni-

cal authors of the period), historical (novels from each decade), generic (from the major sub-genres of nineteenth-century fiction), sociological (set in rural, urban, and mixed locales), and technical (narrated in first-person and third-person form). The novels, as well as important metadata we assigned to them (the perspective and setting), are shown in Table 1. We define *urban* to mean set in a metropolitan zone, characterized by multiple forms of labor (not just agricultural). Here, social relations are largely financial or commercial in character. We conversely define *rural* to describe texts that are set in a country or village zone, where agriculture is the primary activity, and where land-owning, non-productive, rent-collecting gentry are socially predominant. Social relations here are still modeled on feudalism (relations of peasant-lord loyalty and family tie) rather than the commercial cash nexus. We also explored other properties of the texts, such as literary genre, but focus on the results found with setting and perspective. We obtained electronic encodings of the texts from Project Gutenberg. All told, these texts total more than 10 million words.

We assembled this representative corpus in order to test two hypotheses, which are derived from the aforementioned theories:

1. That there is an inverse correlation between the amount of dialogue in a novel and the number of characters in that novel. One basic, shared assumption of these theorists is that as the network of characters expands– as, in Moretti's words, a quantitative change becomes qualitative– the importance, and in fact amount, of dialogue decreases. With a method for extracting conversation from a large corpus of texts, it is possible to test this hypothesis against a wide range of data.

2. That a significant difference in the nineteenth-century novel's representation of social interaction is geographical: novels set in urban environments depict a complex but loose social network, in which numerous characters share little conversational interaction, while novels set in rural environments inhabit more tightly bound social networks, with fewer characters sharing much more conversational interaction. This hypothesis is based on the contrast between Williams's rural "knowable communities" and the sprawling, populous, less conversational urban fictions or Moretti's and Eagleton's analyses. If true, it would suggest that the inverse relationship of hypothesis #1 (more characters means less conversation) can be correlated to, and perhaps even caused by, the geography of a novel's setting. The claims about novelistic geography and social interaction have usually been based on comparisons of a selected few novelists (Jane Austen and Charles Dickens preeminently). Do they remain valid when tested against a larger corpus?

4 Extracting Conversational Networks from Literature

In order to test these hypotheses, we developed a novel approach to extracting social networks from literary texts themselves, building on existing analysis tools. We defined "social network" as "conversational network" for purposes of evaluating these literary theories. In a conversational network, vertices represent characters (assumed to be named entities) and edges indicate at least one instance of dialogue interaction between two characters over the course of the novel. The weight of each edge is proportional to the amount of interaction. We define a conversation as a continuous span of narrative time featuring a set of characters in which the following conditions are met:

1. The characters are in the same place at the same time;

2. The characters take turns speaking; and

3. The characters are mutually aware of each other and each character's speech is mutually intended for the other to hear.

In the following subsections, we discuss the methods we devised for the three problems in text processing invoked by this approach: identifying the characters present in a literary text, assigning a "speaker" (if any) to each instance of quoted speech from among those characters, and constructing a social network by detecting conversations from the set of dialogue acts.

4.1 Character Identification

The first challenge was to identify the candidate speakers by "chunking" names (such as *Mr. Holmes*) from the text. We processed each novel

with the Stanford NER tagger (Finkel et al., 2005) and extracted noun phrases that were categorized as persons or organizations. We then clustered the noun phrases into coreferents for the same entity (person or organization). The clustering process is as follows:

1. For each named entity, we generate variations on the name that we would expect to see in a coreferent. Each variation omits certain parts of multi-word names, respecting titles and first/last name distinctions, similar to work by Davis et al. (2003). For example, *Mr. Sherlock Holmes* may refer to the same character as *Mr. Holmes*, *Sherlock Holmes*, *Sherlock* and *Holmes*.

2. For each named entity, we compile a list of other named entities that may be coreferents, either because they are identical or because one is an expected variation on the other.

3. We then match each named entity to the most recent of its possible coreferents. In aggregate, this creates a cluster of mentions for each character.

We also pre-processed the texts to normalize formatting, detect headings and chapter breaks, remove metadata, and identify likely instances of quoted speech (that is, mark up spans of text that fall between quotation marks, assumed to be a superset of the quoted speech present in the text).

4.2 Quoted Speech Attribution

In order to programmatically assign a speaker to each instance of quoted speech, we applied a high-precision subset of a general approach we describe elsewhere (Elson and McKeown, 2010). The first step of this approach was to compile a separate training and testing corpus of literary texts from British, American and Russian authors of the nineteenth and twentieth centuries. The training corpus consisted of about 111,000 words including 3,176 instances of quoted speech. To obtain gold-standard annotations, we conducted an online survey via Amazon's Mechanical Turk program. For each quote, we asked three annotators to independently choose a speaker from the list of contextual candidates– or, choose "spoken by an unlisted character" if the answer was not available, or "not spoken by any character" for non-dialogue cases such as sneer quotes.

We divided this corpus into training and testing sets, and used the training set to develop a categorizer that assigned one of five syntactic categories to each quote. For example, if a quote is followed by a verb that indicates verbal expression (such as "said"), and then a character mention, a category called *Character trigram* is assigned to the quote. The fifth category is a catch-all for quotes that do not fall into the other four. In many cases, the answer can be reliably determined based solely on its syntactic category. For instance, in the *Character trigram* category, the mentioned character is the quote's speaker in 99% of both the training and testing sets.

In all, we were able to determine the speaker of 57% of the testing set with 96% accuracy just on the basis of syntactic categorization. This is the technique we used to construct our conversational networks. In another study, we applied machine learning tools to the data (one model for each syntactic category) and achieved an overall accuracy of 83% over the entire test set (Elson and McKeown, 2010). The other 43% of quotes are left here as "unknown" speakers; however, in the present study, we are interested in *conversations* rather than individual quotes. Each conversation is likely to consist of multiple quotes by each speaker, increasing the chances of detecting the interaction. Moreover, this design decision emphasizes the precision of the social networks over their recall. This tilts "in favor" of hypothesis #1 (that there are fewer social interactions in larger communities); however, we shall see that despite the emphasis of precision over recall, we identify a sufficient mass of interactions in the texts to constitute evidence against this hypothesis.

4.3 Constructing social networks

We then applied the results from our character identification and quoted speech attribution methods toward the construction of conversational networks from literature. We derived one network from each text in our corpus.

We first assigned vertices to character entities that are mentioned repeatedly throughout the novel. Coreferents for the same name (such as *Mr. Darcy* and *Darcy*) were grouped into the same vertex. We found that a network that included incidental or single-mention named entities became too noisy to function effectively, so we filtered out the entities that are mentioned fewer than three

times in the novel or are responsible for less than 1% of the named entity mentions in the novel.

We assigned undirected edges between vertices that represent *adjacency* in quoted speech fragments. Specifically, we set the weight of each undirected edge between two character vertices to the total length, in words, of all quotes that either character speaks from among all pairs of adjacent quotes in which they both speak– implying face to face conversation. We empirically determined that the most accurate definition of "adjacency" is one where the two characters' quotes fall within 300 words of one another with no attributed quotes in between. When such an adjacency is found, the length of the quote is added to the edge weight, under the hypothesis that the significance of the relationship between two individuals is proportional to the length of the dialogue that they exchange. Finally, we normalized each edge's weight by the length of the novel.

An example network, automatically constructed in this manner from Jane Austen's *Mansfield Park*, is shown in Figure 1. The width of each vertex is drawn to be proportional to the character's share of all the named entity mentions in the book (so that protagonists, who are mentioned frequently, appear in larger ovals). The width of each edge is drawn to be proportional to its weight (total conversation length).

We also experimented with two alternate methods for identifying edges, for purposes of a baseline:

1. The "correlation" method divides the text into 10-paragraph segments and counts the number of mentions of each character in each segment (excluding mentions inside quoted speech). It then computes the Pearson product-moment correlation coefficient for the distributions of mentions for each pair of characters. These coefficients are used for the edge weights. Characters that tend to appear together in the same areas of the novel are taken to be more socially connected, and have a higher edge weight.

2. The "spoken mention" method counts occurrences when one character refers to another in his or her quoted speech. These counts, normalized by the length of the text, are used as edge weights. The intuition is that characters who refer to one another are likely to be in conversation.

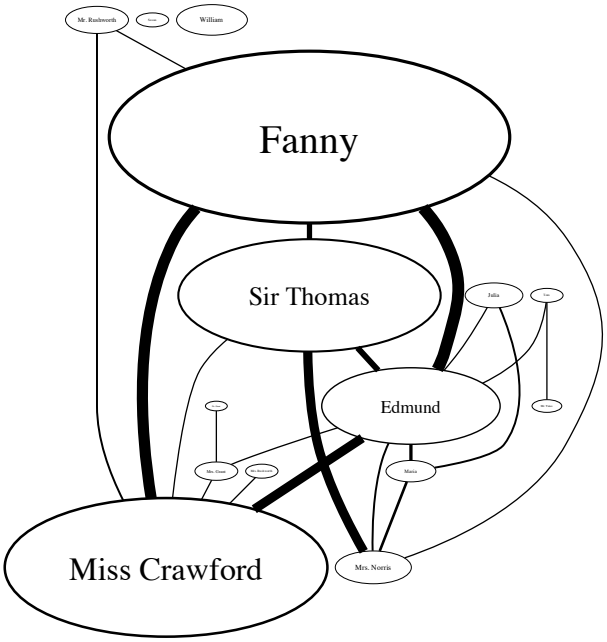


Figure 1: Automatically extracted conversation network for Jane Austen's *Mansfield Park*.

4.4 Evaluation

To check the accuracy of our method for extracting conversational networks, we conducted an evaluation involving four of the novels (*The Sign of the Four*, *Emma*, *David Copperfield* and *The Portrait of a Lady*). We did not use these texts when developing our method for identifying conversations. For each book, we randomly selected 4-5 chapters from among those with significant amounts of quoted speech, so that all excerpts from each novel amounted to at least 10,000 words. We then asked three annotators to identity all the conversations that occur in all 44,000 words. We requested that the annotators include both direct and indirect (unquoted) speech, and define "conversation" as in the beginning of Section 4, but exclude "retold" conversations (those that occur within other dialogue).

We processed the annotation results by breaking down each multi-way conversation into all of its unique two-character interactions (for example, a conversation between four people indicates six bilateral interactions). To calculate inter-annotator agreement, we first compiled a list of all possible interactions between all characters in each text. In this model, each annotator contributed a set of "yes" or "no" decisions, one for every character pair. We then applied the kappa measurement for agreement in a binary classification problem (Co-

Method	Precision	Recall	F
Speech adjacency	.95	.51	.67
Correlation	.21	.65	.31
Spoken-mention	.45	.49	.47

Table 2: Precision, recall, and F-measure of three methods for detecting bilateral conversations in literary texts.

hen, 1960). In 95% of character pairs, annotators were unanimous, which is a high agreement of $k = .82$.

The precision and recall of our method for detecting conversations is shown in Table 2. Precision was .95; this indicates that we can be confident in the specificity of the conversational networks that we automatically construct. Recall was .51, indicating a sensitivity of slightly more than half. There were several reasons that we did not detect the missing links, including indirect speech, quotes attributed to anaphoras or coreferents, and "diffuse" conversations in which the characters do not speak in turn with one another.

To calculate precision and recall for the two baseline social networks, we set a threshold t to derive a binary prediction from the continuous edge weights. The precision and recall values shown for the baselines in Table 2 represent the highest performance we achieved by varying t between 0 and 1 (maximizing F-measure over t). Both baselines performed significantly worse in precision and F-measure than our quoted speech adjacency method for detecting conversations.

5 Data Analysis

5.1 Feature extraction

We extracted features from the conversational networks that emphasize the complexity of the social interactions found in each novel:

1. The number of characters and the number of speaking characters

2. The variance of the distribution of quoted speech (specifically, the proportion of quotes spoken by the n most frequent speakers, for $1 \leq n \leq 5$)

3. The number of quotes, and proportion of words in the novel that are quoted speech

4. The number of 3-cliques and 4-cliques in the social network

5. The *average degree* of the graph, defined as

$$\frac{\sum_{v \in V} |E_v|}{|V|} = \frac{2|E|}{|V|} \quad (1)$$

where $|E_v|$ is the number of edges incident on a vertex v, and $|V|$ is the number of vertices. In other words, this determines the average number of characters connected to each character in the conversational network ("with how many people on average does a character converse?").

6. A variation on *graph density* that normalizes the average degree feature by the number of characters:

$$\frac{\sum_{v \in V} |E_v|}{|V|(|V|-1)} = \frac{2|E|}{|V|(|V|-1)} \quad (2)$$

By dividing again by $|V| - 1$, we use this as a metric for the overall connectedness of the graph: "with what *percent* of the entire network (besides herself) does each character converse, on average?" The weight of the edge, as long as it is greater than 0, does not affect either the network's average degree or graph density.

5.2 Results

We derived results from the data in two ways. First, we examined the strengths of the correlations between the features that we extracted (for example, between number of character vertices and the average degree of each vertex). We used Pearson's product-moment correlation coefficient in these calculations. Second, we compared the extracted features to the metadata we previously assigned to each text (e.g., urban vs. rural).

Hypothesis #1, which we described in Section 3, claims that there is an inverse correlation between the amount of dialogue in a nineteenth-century novel and the number of characters in that novel. We did not find this to be the case. Rather, we found a weak but positive correlation (r=.16) between the number of quotes in a novel and the number of characters (normalizing the quote count for text length). There was a stronger positive correlation (r=.50) between the number of unique speakers (those characters who speak at least once) and the normalized number of quotes, suggesting that larger networks have more conversations than smaller ones. But because the first

correlation is weak, we investigated whether further analysis could identify other evidence that confirms or contradicts the hypothesis.

Another way to interpret hypothesis #1 is that social networks with more characters tend to break apart and be less connected. However, we found the opposite to be true. The correlation between the number of characters in each graph and the average degree (number of conversation partners) for each character was a positive, moderately strong r=.42. This is not a given; a network can easily, for example, break into minimally connected or mutually exclusive subnetworks when more characters are involved. Instead, we found that networks tend to stay close-knit regardless of their size: even the density of the graph (the percentage of the community that each character talks to) grows with the total population size at r=.30. Moreover, as the population of *speakers* grows, the density is likely to increase at r=.49. A higher number of characters (speaking or non-speaking) is also correlated with a higher *rate* of 3-cliques per character (r=.38), as well as with a more balanced distribution of dialogue (the share of dialogue spoken by the top three speakers decreases at r=−.61). This evidence suggests that in nineteenth-century British literature, it is the small communities, rather than the large ones, that tend to be disconnected.

Hypothesis #2, meanwhile, posited that a novel's setting (urban or rural) would have an effect on the structure of its social network. After defining "social network" as a conversational network, we did not find this to be the case. Surprisingly, the numbers of characters and speakers found in the urban novel were *not* significantly greater than those found in the rural novel. Moreover, each of the features we extracted, such as the rate of cliques, average degree, density, and rate of characters' mentions of other characters, did not change in a statistically significant manner between the two genres. For example, Figure 2 shows the mean over all texts of each network's average degree, with confidence intervals, separated by setting into urban and rural. The increase in degree seen in urban texts is not significant.

Rather, the only type of metadata variable that *did* impact the average degree with any significance was the text's perspective. Figure 2 also separates texts into first- and third-person tellings and shows the means and confidence intervals for the

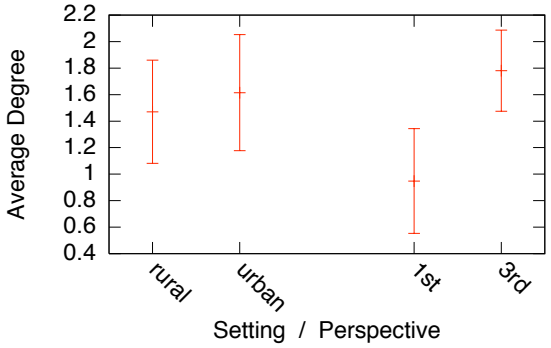


Figure 2: The average degree for each character as a function of the novel's setting and its perspective.

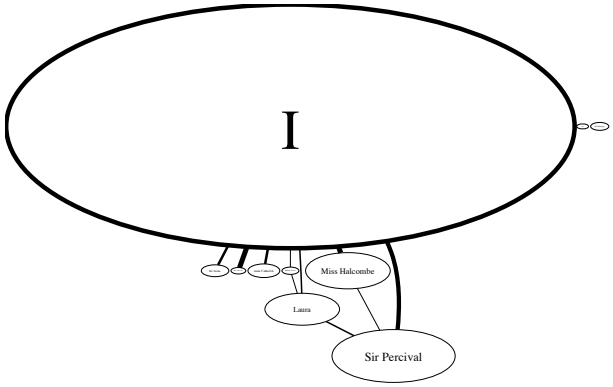


Figure 3: Conversational networks for first-person novels like Collins's *The Woman in White* are less connected due to the structure imposed by the perspective.

average degree measure. Stories told in the third person had much more connected networks than stories told in the first person: not only did the average degree increase with statistical significance (by the homoscedastic t-test to $p < .005$), so too did the graph density ($p < .05$) and the rate of 3-cliques per character ($p < .05$).

We believe the reason for this can be intuited with a visual inspection of a first-person graph. Figure 3 shows the conversational network extracted for Collins's *The Woman in White*, which is told in the first person. Not surprisingly, the most oft-repeated named entity in the text is *I*, referring to the narrator. More surprising is the *lack* of conversation connections between the auxiliary characters. The story's structure revolves around the narrator and each character is understood in terms of his or her relationship to the narrator. Private conversations between auxiliary characters would not include the narrator, and thus do not appear in a

first-hand account. An "omniscient" third person narrator, by contrast, can eavesdrop on any pair of characters conversing. This highlights the importance of detecting reported and indirect speech in future work, as a first-person narrator may hear about other connections without witnessing them.

6 Literary Interpretation of Results

Our data, therefore, markedly do not confirm hypothesis #1. They also suggest, in relation to hypothesis #2 (also not confirmed by the data), a strong reason why.

One of the basic assumptions behind hypothesis #2– that urban novels contain more characters, mirroring the masses of nineteenth-century cities– is not borne out by our data. Our results do, however, strongly correlate a point of view (third-person narration) with more frequently connected characters, implying tighter and more talkative social networks.

We would propose that this suggests that the form of a given novel– the standpoint of the narrative voice, whether the voice is "omniscient" or not– is far more determinative of the kind of social network described in the novel than where it is set or even the number of characters involved. Whereas standard accounts of nineteenth-century fiction, following Bakhtin's notion of the "chronotope," emphasize the content of the novel as determinative (where it is set, whether the novel fits within a genre of "village" or "urban" fiction), we have found that content to be surprisingly irrelevant to the shape of social networks within. Bakhtin's influential theory, and its detailed reworkings by Williams, Moretti, and others, suggests that as the novel becomes more urban, more centered in (and interested in) populous urban settings, the novel's form changes to accommodate the looser, more populated, less conversational networks of city life. Our data suggests the opposite: that the "urban novel" is not as strongly distinctive a form as has been asserted, and that in fact it can look much like the village fictions of the century, as long as the same method of narration is used.

This conclusion leads to some further considerations. We are suggesting that the important element of social networks in nineteenth-century fiction is not where the networks are set, but from what standpoint they are imagined or narrated. Narrative voice, that is, trumps setting.

7 Conclusion

In this paper, we presented a method for characterizing a text of literary fiction by extracting the network of social conversations that occur between its characters. This allowed us to take a systematic and wide look at a large corpus of texts, an approach which complements the narrower and deeper analysis performed by literary scholars and can provide evidence for or against some of their claims. In particular, we described a high-precision method for detecting face-to-face conversations between two named characters in a novel, and showed that as the number of characters in a novel grows, so too do the cohesion, interconnectedness and balance of their social network. In addition, we showed that the form of the novel (first- or third-person) is a stronger predictor of these features than the setting (urban or rural). Our results thus far suggest further review of our methods, our corpus and our results for more insights into the social networks found in this and other genres of fiction.

8 Acknowledgments

This material is based on research supported in part by the U.S. National Science Foundation (NSF) under IIS-0935360. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

Mikhail Bakhtin. 1981. Forms of time and of the chronotope in the novel. In Trans. Michael Holquist and Caryl Emerson, editors, *The Dialogic Imagination: Four Essays*, pages 84–258. University of Texas Press, Austin.

John Burrows. 2004. Textual analysis. In Susan Schreibman, Ray Siemens, and John Unsworth, editors, *A Companion to Digital Humanities*. Blackwell, Oxford.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *In Proceedings of the 46th Annual Meeting of the Association of Com- putational Linguistics (ACL-08)*, pages 789–797, Columbus, Ohio.

Wendy K. Tam Cho and James H. Fowler. 2010. Legislative success in a small world: Social network analysis and the dynamics of congressional legislation. *The Journal of Politics*, 72(1):124–135.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Peter T. Davis, David K. Elson, and Judith L. Klavans. 2003. Methods for precise named entity matching in digital collections. In *Proceedings of the Third ACM/IEEE Joint Conference on Digital Libraries (JCDL '03)*, Houston, Texas.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 837–840, Lisbon.

Terry Eagleton. 2005. *The English Novel: An Introduction*. Blackwell, Oxford.

David K. Elson and Kathleen R. McKeown. 2010. Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, Atlanta, Georgia.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370.

Anatoliy Gruzd and Caroline Haythornthwaite. 2008. Automated discovery and analysis of social networks from threaded discussions. In *International Network of Social Network Analysis (INSNA) Conference*, St. Pete Beach, Florida.

Harry Halpin, Johanna D. Moore, and Judy Robertson. 2004. Automatic analysis of plot for story rewriting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, Barcelona.

John Lee. 2007. A computational model of text reuse in ancient literary texts. In *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 472–479, Prague.

Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanual. 2007. Topic and role discovery in social networks with experiments on enron and academic email. *Journal of Artificial Intelligence Research*, 30:249–272.

Franco Moretti. 1999. *Atlas of the European Novel, 1800-1900*. Verso, London.

Franco Moretti. 2005. *Graphs, Maps, Trees: Abstract Models for a Literary History*. Verso, London.

Frederick Mostellar and David L. Wallace. 1984. *Applied Bayesian and Classical Inference: The Case of The Federalist Papers*. Springer, New York.

Raymond Williams. 1975. *The Country and The City*. Oxford University Press, Oxford.

Pseudo-word for Phrase-based Machine Translation

Xiangyu Duan **Min Zhang** **Haizhou Li**
Institute for Infocomm Research, A-STAR, Singapore
{Xduan, mzhang, hli}@i2r.a-star.edu.sg

Abstract

The pipeline of most Phrase-Based Statistical Machine Translation (PB-SMT) systems starts from automatically word aligned parallel corpus. But word appears to be too fine-grained in some cases such as non-compositional phrasal equivalences, where no clear word alignments exist. Using words as inputs to PB-SMT pipeline has inborn deficiency. This paper proposes pseudo-word as a new start point for PB-SMT pipeline. Pseudo-word is a kind of basic multi-word expression that characterizes minimal sequence of consecutive words in sense of translation. By casting pseudo-word searching problem into a parsing framework, we search for pseudo-words in a monolingual way and a bilingual synchronous way. Experiments show that pseudo-word significantly outperforms word for PB-SMT model in both travel translation domain and news translation domain.

1 Introduction

The pipeline of most Phrase-Based Statistical Machine Translation (PB-SMT) systems starts from automatically word aligned parallel corpus generated from word-based models (Brown et al., 1993), proceeds with step of induction of phrase table (Koehn et al., 2003) or synchronous grammar (Chiang, 2007) and with model weights tuning step. Words are taken as inputs to PB-SMT at the very beginning of the pipeline. But there is a deficiency in such manner that word is too fine-grained in some cases such as non-compositional phrasal equivalences, where clear word alignments do not exist. For example in Chinese-to-English translation, "想" and "would like to" constitute a *1*-to-*n* phrasal equivalence, "多少钱" and "how much is it" constitute a *m*-to-*n* phrasal equivalence. No clear word alignments are there in such phrasal equivalences. Moreover, should basic translational unit be word or coarse-grained multi-word is an open problem for optimizing SMT models.

Some researchers have explored coarse-grained translational unit for machine translation. Marcu and Wong (2002) attempted to directly learn phrasal alignments instead of word alignments. But computational complexity is prohibitively high for the exponentially large number of decompositions of a sentence pair into phrase pairs. Cherry and Lin (2007) and Zhang et al. (2008) used synchronous ITG (Wu, 1997) and constraints to find non-compositional phrasal equivalences, but they suffered from intractable estimation problem. Blunsom et al. (2008; 2009) induced phrasal synchronous grammar, which aimed at finding hierarchical phrasal equivalences.

Another direction of questioning word as basic translational unit is to directly question word segmentation on languages where word boundaries are not orthographically marked. In Chinese-to-English translation task where Chinese word boundaries are not marked, Xu et al. (2004) used word aligner to build a Chinese dictionary to re-segment Chinese sentence. Xu et al. (2008) used a Bayesian semi-supervised method that combines Chinese word segmentation model and Chinese-to-English translation model to derive a Chinese segmentation suitable for machine translation. There are also researches focusing on the impact of various segmentation tools on machine translation (Ma et al. 2007; Chang et al. 2008; Zhang et al. 2008). Since there are many *1*-to-*n* phrasal equivalences in Chinese-to-English translation (Ma and Way. 2009), only focusing on Chinese word as basic translational unit is not adequate to model *1*-to-*n* translations. Ma and Way (2009) tackle this problem by using word aligner to bootstrap bilingual segmentation suitable for machine translation. Lambert and Banchs (2005) detect bilingual multi-word ex-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 148–156,
Uppsala, Sweden, 11-16 July 2010.

pressions by monotonically segmenting a given Spanish-English sentence pair into bilingual units, where word aligner is also used.

IBM model 3, 4, 5 (Brown et al., 1993) and Deng and Byrne (2005) are another kind of related works that allow *1*-to-*n* alignments, but they rarely questioned if such alignments exist in word units level, that is, they rarely questioned word as basic translational unit. Moreover, *m*-to-*n* alignments were not modeled.

This paper focuses on determining the basic translational units on both language sides without using word aligner before feeding them into PB-SMT pipeline. We call such basic translational unit as pseudo-word to differentiate with word. Pseudo-word is a kind of multi-word expression (includes both unary word and multi-word). Pseudo-word searching problem is the same to decomposition of a given sentence into pseudo-words. We assume that such decomposition is in the Gibbs distribution. We use a measurement, which characterizes pseudo-word as minimal sequence of consecutive words in sense of translation, as potential function in Gibbs distribution. Note that the number of decomposition of one sentence into pseudo-words grows exponentially with sentence length. By fitting decomposition problem into parsing framework, we can find optimal pseudo-word sequence in polynomial time. Then we feed pseudo-words into PB-SMT pipeline, and find that pseudo-words as basic translational units improve translation performance over words as basic translational units. Further experiments of removing the power of higher order language model and longer max phrase length, which are inherent in pseudo-words, show that pseudo-words still improve translational performance significantly over unary words.

This paper is structured as follows: In section 2, we define the task of searching for pseudo-words and its solution. We present experimental results and analyses of using pseudo-words in PB-SMT model in section 3. The conclusion is presented at section 4.

2 Searching for Pseudo-words

Pseudo-word searching problem is equal to decomposition of a given sentence into pseudo-words. We assume that the distribution of such decomposition is in the form of Gibbs distribution as below:

$$P(Y \mid X) = \frac{1}{Z_X} \exp(\sum_k Sig_{y_k}) \quad (1)$$

where X denotes the sentence, Y denotes a decomposition of X. *Sig* function acts as potential function on each multi-word y_k, and Z_X acts as partition function. Note that the number of y_k is not fixed given X because X can be decomposed into various number of multi-words.

Given X, Z_X is fixed, so searching for optimal decomposition is as below:

$$\hat{Y} = \underset{Y}{ARGMAX}\, P(Y \mid X) = \underset{Y_1^K}{ARGMAX} \sum_k Sig_{y_k} \quad (2)$$

where Y_1^K denotes K multi-word units from decomposition of X. A multi-word sequence with maximal sum of *Sig* function values is the search target — pseudo-word sequence. From (2) we can see that *Sig* function is vital for pseudo-word searching. In this paper *Sig* function calculates sequence significance which is proposed to characterize pseudo-word as minimal sequence of consecutive words in sense of translation. The detail of sequence significance is described in the following section.

2.1 Sequence Significance

Two kinds of definitions of sequence significance are proposed. One is monolingual sequence significance. X and Y are monolingual sentence and monolingual multi-words respectively in this monolingual scenario. The other is bilingual sequence significance. X and Y are sentence pair and multi-word pairs respectively in this bilingual scenario.

2.1.1 Monolingual Sequence Significance

Given a sentence $w_1, \ldots, w_n$, where w_i denotes unary word, monolingual sequence significance is defined as:

$$Sig_{i,j} = \frac{Freq_{i,j}}{Freq_{i-1,j+1}} \quad (3)$$

where $Freq_{i,j}$ ($i \leq j$) represents frequency of word sequence $w_i, \ldots, w_j$ in the corpus, $Sig_{i,j}$ represents monolingual sequence significance of a word sequence $w_i, \ldots, w_j$. We also denote word sequence $w_i, \ldots, w_j$ as $span[i, j]$, whole sentence as $span[1, n]$. Each span is also a multi-word expression.

Monolingual sequence significance of $span[i, j]$ is proportional to $span[i, j]$'s frequency, while is inversely proportion to frequency of expanded span ($span[i-1, j+1]$). Such definition characterizes minimal sequence of consecutive words which we are looking for. Our target is to find pseudo-word sequence which has maximal sum of spans' significances:

$$pw_1^K = \underset{span_1^K}{ARGMAX} \sum_{k=1}^{K} Sig_{span_k} \quad (4)$$

where *pw* denotes pseudo-word, *K* is equal to or less than sentence's length. $span_k$ is the *k*th span of *K* spans $span_1^K$. Equation (4) is the rewrite of equation (2) in monolingual scenario. Searching for pseudo-words pw_1^K is the same to finding optimal segmentation of a sentence into *K* segments $span_1^K$ (*K* is a variable too). Details of searching algorithm are described in section 2.2.1.

We firstly search for monolingual pseudo-words on source and target side individually. Then we apply word alignment techniques to build pseudo-word alignments. We argue that word alignment techniques will work fine if non-existent word alignments in such as non-compositional phrasal equivalences have been filtered by pseudo-words.

2.1.2 Bilingual Sequence Significance

Bilingual sequence significance is proposed to characterize pseudo-word pairs. Co-occurrence of sequences on both language sides is used to define bilingual sequence significance. Given a bilingual sequence pair: *span-pair*[i_s, j_s, i_t, j_t] (source side *span*[i_s, j_s] and target side *span*[i_t, j_t]), bilingual sequence significance is defined as below:

$$Sig_{i_s, j_s, i_t, j_t} = \frac{Freq_{i_s, j_s, i_t, j_t}}{Freq_{i_s-1, j_s+1, i_t-1, j_t+1}} \quad (5)$$

where *Freq* denotes the frequency of a span-pair. Bilingual sequence significance is an extension of monolingual sequence significance. Its value is proportional to frequency of *span-pair*[i_s, j_s, i_t, j_t], while is inversely proportional to frequency of expanded *span-pair*[i_s-1, j_s+1, i_t-1, j_t+1]. Pseudo-word pairs of one sentence pair are such pairs that maximize the sum of *span-pairs*' bilingual sequence significances:

$$pwp_1^K = \underset{span-pair_1^K}{ARGMAX} \sum_{k=1}^{K} Sig_{span-pair_k} \quad (6)$$

pwp represents pseudo-word pair. Equation (6) is the rewrite of equation (2) in bilingual scenario. Searching for pseudo-word pairs pwp_1^K is equal to bilingual segmentation of a sentence pair into optimal $span\text{-}pair_1^K$. Details of searching algorithm are presented in section 2.2.2.

2.2 Algorithms of Searching for Pseudo-words

Pseudo-word searching problem is equal to decomposition of a sentence into pseudo-words. But the number of possible decompositions of the sentence grows exponentially with the sentence length in both monolingual scenario and bilingual scenario. By casting such decomposition problem into parsing framework, we can find pseudo-word sequence in polynomial time. According to the two scenarios, searching for pseudo-words can be performed in a monolingual way and a synchronous way. Details of the two kinds of searching algorithms are described in the following two sections.

2.2.1 Algorithm of Searching for Monolingual Pseudo-words (SMP)

Searching for monolingual pseudo-words is based on the computation of monolingual sequence significance. Figure 1 presents the search algorithm. It is performed in a way similar to CKY (Cocke-Kasami-Younger) parser.

```
Initialization: W_{i,i} = Sig_{i,i};
                W_{i,j} = 0,  (i≠j);
1: for d = 2 ... n do
2:    for all i, j s.t. j-i=d-1 do
3:       for k = i ... j – 1 do
4:          v = W_{i,k} + W_{k+1,j}
5:          if v > W_{i,j} then
6:             W_{i,j} = v;
7:       u = Sig_{i,j}
8:       if u > W_{i,j} then
9:          W_{i,j} = u;
```

Figure 1. Algorithm of searching for monolingual pseudo-words (SMP).

In this algorithm, $W_{i,j}$ records maximal sum of monolingual sequence significances of sub spans of *span*[*i, j*]. During initialization, $W_{i,i}$ is initialized as $Sig_{i,i}$ (note that this sequence is word w_i only). For all spans that have more than one word ($i \neq j$), $W_{i,j}$ is initialized as zero.

In the main algorithm, *d* represents span's length, ranging from *2* to *n*, *i* represents start position of a span, *j* represents end position of a span, *k* represents decomposition position of *span*[*i,j*]. For *span*[*i, j*], $W_{i,j}$ is updated if higher sum of monolingual sequence significances is found.

The algorithm is performed in a bottom-up way. Small span's computation is first. After maximal sum of significances is found in small spans, big span's computation, which uses small spans' maximal sum, is continued. Maximal sum of significances for whole sentence ($W_{1,n}$, *n* is sentence's length) is guaranteed in this way, and optimal decomposition is obtained correspondingly.

The method of fitting the decomposition problem into CKY parsing framework is located at steps 7-9. After steps 3-6, all possible decompositions of *span*[*i, j*] are explored and $W_{i,j}$ of optimal decomposition of *span*[*i, j*] is recorded. Then monolingual sequence significance $Sig_{i,j}$ of *span*[*i, j*] is computed at step 7, and it is compared to $W_{i,j}$ at step 8. Update of $W_{i,j}$ is taken at step 9 if $Sig_{i,j}$ is bigger than $W_{i,j}$, which indicates that *span*[*i*, *j*] is non-decomposable. Thus whether *span*[*i*, *j*] should be non-decomposable or not is decided through steps 7-9.

2.2.2 Algorithm of Synchronous Searching for Pseudo-words (SSP)

Synchronous searching for pseudo-words utilizes bilingual sequence significance. Figure 2 presents the search algorithm. It is similar to ITG (Wu, 1997), except that it has no production rules and non-terminal nodes of a synchronous grammar. What it cares about is the span-pairs that maximize the sum of bilingual sequence significances.

Initialization: **if** $i_s = j_s$ or $i_t = j_t$ **then**
$W_{i_s,j_s,i_t,j_t} = Sig_{i_s,j_s,i_t,j_t}$;
else
$W_{i_s,j_s,i_t,j_t} = 0$;

1: **for** $d_s = 2 \ldots n_s$, $d_t = 2 \ldots n_t$ **do**
2: **for** all i_s, j_s, i_t, j_t s.t. $j_s - i_s = d_s - 1$ and $j_t - i_t = d_t - 1$ **do**
3: **for** $k_s = i_s \ldots j_s - 1$, $k_t = i_t \ldots j_t - 1$ **do**
4: $v = max\{W_{i_s,k_s,i_t,k_t} + W_{k_s+1,j_s,k_t+1,j_t}, W_{i_s,k_s,k_t+1,j_t} + W_{k_s+1,j_s,i_t,k_t}\}$
5: **if** $v > W_{i_s,j_s,i_t,j_t}$ **then**
6: $W_{i_s,j_s,i_t,j_t} = v$;
7: $u = Sig_{i_s,j_s,i_t,j_t}$
8: **if** $u > W_{i_s,j_s,i_t,j_t}$ **then**
9: $W_{i_s,j_s,i_t,j_t} = u$;

Figure 2. Algorithm of Synchronous Searching for Pseudo-words(SSP).

In the algorithm, W_{i_s,j_s,i_t,j_t} records maximal sum of bilingual sequence significances of sub span-pairs of *span-pair*[i_s, j_s, i_t, j_t]. For *1*-to-*m* span-pairs, *W*s are initialized as bilingual sequence significances of such span-pairs. For other span-pairs, *W*s are initialized as zero.

In the main algorithm, d_s/d_t denotes the length of a span on source/target side, ranging from *2* to n_s/n_t (source/target sentence's length). i_s/i_t is the start position of a span-pair on source/target side, j_s/j_t is the end position of a span-pair on source/target side, k_s/k_t is the decomposition position of a *span-pair*[i_s, j_s, i_t, j_t] on source/target side.

Update steps in Figure 2 are similar to that of Figure 1, except that the update is about span-pairs, not monolingual spans. Reversed and non-reversed alignments inside a span-pair are compared at step 4. For *span-pair*[i_s, j_s, i_t, j_t], W_{i_s,j_s,i_t,j_t} is updated at step 6 if higher sum of bilingual sequence significances is found.

Fitting the bilingually searching for pseudo-words into ITG framework is located at steps 7-9. Steps 3-6 have explored all possible decompositions of *span-pair*[i_s, j_s, i_t, j_t] and have recorded maximal W_{i_s,j_s,i_t,j_t} of these decompositions. Then bilingual sequence significance of *span-pair*[i_s, j_s, i_t, j_t] is computed at step 7. It is compared to W_{i_s,j_s,i_t,j_t} at step 8. Update is taken at step 9 if bilingual sequence significance of *span-pair*[i_s, j_s, i_t, j_t] is bigger than W_{i_s,j_s,i_t,j_t}, which indicates that *span-pair*[i_s, j_s, i_t, j_t] is non-decomposable. Whether the *span-pair*[i_s, j_s, i_t, j_t] should be non-decomposable or not is decided through steps 7-9.

In addition to the initialization step, all span-pairs' bilingual sequence significances are computed. Maximal sum of bilingual sequence significances for one sentence pair is guaranteed through this bottom-up way, and the optimal decomposition of the sentence pair is obtained correspondingly.

- **Algorithm of Excluded Synchronous Searching for Pseudo-words (ESSP)**

The algorithm of SSP in Figure 2 explores all span-pairs, but it neglects NULL alignments, where words and "empty" word are aligned. In fact, SSP requires that all parts of a sentence pair should be aligned. This requirement is too strong because NULL alignments are very common in many language pairs. In SSP, words that should be aligned to "empty" word are programmed to be aligned to real words.

Unlike most word alignment methods (Och and Ney, 2003) that add "empty" word to account for NULL alignment entries, we propose a method to naturally exclude such NULL alignments. We call this method as Excluded Synchronous Searching for Pseudo-words (ESSP).

The main difference between ESSP and SSP is in steps 3-6 in Figure 3. We illustrate Figure 3's span-pair configuration in Figure 4.

Initialization: **if** $i_s = j_s$ or $i_t = j_t$ **then**

$W_{i_s,j_s,i_t,j_t} = Sig_{i_s,j_s,i_t,j_t}$;

else

$W_{i_s,j_s,i_t,j_t} = 0$;

1: **for** $d_s = 2 \ldots n_s$, $d_t = 2 \ldots n_t$ **do**
2: **for** all i_s, j_s, i_t, j_t s.t. $j_s - i_s = d_s - 1$ and $j_t - i_t = d_t - 1$ **do**
3: **for** $k_{s1} = i_s + 1 \ldots j_s$, $k_{s2} = k_{s1} - 1 \ldots j_s - 1$
$k_{t1} = i_t + 1 \ldots j_t$, $k_{t2} = k_{t1} - 1 \ldots j_t - 1$ **do**
4: $v = \max\{W_{i_s,k_{s1}-1,i_t,k_{t1}-1} + W_{k_{s2}+1,j_s,k_{t2}+1,j_t}, W_{i_s,k_{s1}-1,k_{t2}+1,j_t} + W_{k_{s2}+1,j_s,i_t,k_{t1}-1}\}$
5: **if** $v > W_{i_s,j_s,i_t,j_t}$ **then**
6: $W_{i_s,j_s,i_t,j_t} = v$;
7: $u = Sig_{i_s,j_s,i_t,j_t}$
8: **if** $u > W_{i_s,j_s,i_t,j_t}$ **then**
9: $W_{i_s,j_s,i_t,j_t} = u$;

Figure 3. Algorithm of Excluded Synchronous Searching for Pseudo-words (ESSP).

The solid boxes in Figure 4 represent excluded parts of *span-pair*[i_s, j_s, i_t, j_t] in ESSP. Note that, in SSP, there is no excluded part, that is, $k_{s1}=k_{s2}$ and $k_{t1}=k_{t2}$.

We can see that in Figure 4, each monolingual span is configured into three parts, for example: *span*[i_s, k_{s1}-*1*], *span*[k_{s1}, k_{s2}] and *span*[k_{s2}+*1*, j_s] on source language side. k_{s1} and k_{s2} are two new variables gliding between i_s and j_s, *span*[k_{s1}, k_{s2}] is source side excluded part of *span-pair*[i_s, j_s, i_t, j_t]. Bilingual sequence significance is computed only on pairs of blank boxes, solid boxes are excluded in this computation to represent NULL alignment cases.

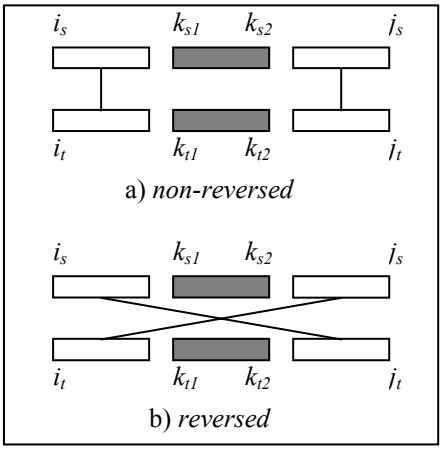


Figure 4. Illustration of excluded configuration.

Note that, in Figure 4, solid box on either language side can be void (i.e., length is zero) if there is no NULL alignment on its side. If all solid boxes are shrunk into void, algorithm of ESSP is the same to SSP.

Generally, span length of NULL alignment is not very long, so we can set a length threshold for NULL alignments, eg. $k_{s2}-k_{s1} \leqslant EL$, where *EL* denotes *Excluded Length* threshold. Computational complexity of the ESSP remains the same to SSP's complexity $O(n_s^3.n_t^3)$, except multiply a constant EL^2.

There is one kind of NULL alignments that ESSP can not consider. Since we limit excluded parts in the middle of a span-pair, the algorithm will end without considering boundary parts of a sentence pair as NULL alignments.

3 Experiments and Results

In our experiments, pseudo-words are fed into PB-SMT pipeline. The pipeline uses GIZA++ model 4 (Brown et al., 1993; Och and Ney, 2003) for pseudo-word alignment, uses Moses (Koehn et al., 2007) as phrase-based decoder, uses the SRI Language Modeling Toolkit to train language model with modified Kneser-Ney smoothing (Kneser and Ney 1995; Chen and Goodman 1998). Note that MERT (Och, 2003) is still on original words of target language. In our experiments, pseudo-word length is limited to no more than six unary words on both sides of the language pair.

We conduct experiments on Chinese-to-English machine translation. Two data sets are adopted, one is small corpus of IWSLT-2008 BTEC task of spoken language translation in travel domain (Paul, 2008), the other is large corpus in news domain, which consists Hong Kong News (LDC2004T08), Sinorama Magazine (LDC2005T10), FBIS (LDC2003E14), Xinhua (LDC2002E18), Chinese News Translation (LDC2005T06), Chinese Treebank (LDC2003E07), Multiple Translation Chinese (LDC2004T07). Table 1 lists statistics of the corpus used in these experiments.

	small		large	
	Ch → En		Ch → En	
Sent.	23k		1,239k	
word	190k	213k	31.7m	35.5m
ASL	8.3	9.2	25.6	28.6

Table 1. Statistics of corpora, "Ch" denotes Chinese, "En" denotes English, "Sent." row is the number of sentence pairs, "word" row is the number of words, "ASL" denotes average sentence length.

For small corpus, we use CSTAR03 as development set, use IWSLT08 official test set for test. A *5*-gram language model is trained on English side of parallel corpus. For large corpus, we use NIST02 as development set, use NIST03 as test set. Xinhua portion of the English Gigaword3 corpus is used together with English side of large corpus to train a *4*-gram language model.

Experimental results are evaluated by case-insensitive BLEU-4 (Papineni et al., 2001). Closest reference sentence length is used for brevity penalty. Additionally, NIST score (Doddington, 2002) and METEOR (Banerjee and Lavie, 2005) are also used to check the consistency of experimental results. Statistical significance in BLEU score differences was tested by paired bootstrap re-sampling (Koehn, 2004).

3.1 Baseline Performance

Our baseline system feeds word into PB-SMT pipeline. We use GIZA++ model 4 for word alignment, use Moses for phrase-based decoding. The setting of language model order for each corpus is not changed. Baseline performances on test sets of small corpus and large corpus are reported in table 2.

	small	Large
BLEU	0.4029	0.3146
NIST	7.0419	8.8462
METEOR	0.5785	0.5335

Table 2. Baseline performances on test sets of small corpus and large corpus.

3.2 Pseudo-word Unpacking

Because pseudo-word is a kind of multi-word expression, it has inborn advantage of higher language model order and longer max phrase length over unary word. To see if such inborn advantage is the main contribution to the performance or not, we unpack pseudo-word into words after GIZA++ aligning. Aligned pseudo-words are unpacked into $m \times n$ word alignments. PB-SMT pipeline is executed thereafter. The advantage of longer max phrase length is removed during phrase extraction, and the advantage of higher order of language model is also removed during decoding since we use language model trained on unary words. Performances of pseudo-word unpacking are reported in section 3.3.1 and 3.4.1. Ma and Way (2009) used the unpacking after phrase extraction, then re-estimated phrase translation probability and lexical reordering model. The advantage of longer max phrase length is still used in their method.

3.3 Pseudo-word Performances on Small Corpus

Table 3 presents performances of SMP, SSP, ESSP on small data set. $pw_{ch}pw_{en}$ denotes that pseudo-words are on both language side of training data, and they are input strings during development and testing, and translations are also pseudo-words, which will be converted to words as final output. $w_{ch}pw_{en}/pw_{ch}w_{en}$ denotes that pseudo-words are adopted only on English/Chinese side of the data set.

We can see from table 3 that, ESSP attains the best performance, while SSP attains the worst performance. This shows that excluding NULL alignments in synchronous searching for pseudo-words is effective. SSP puts overly strong alignment constraints on parallel corpus, which impacts performance dramatically. ESSP is superior to SMP indicating that bilingually motivated searching for pseudo-words is more effective. Both SMP and ESSP outperform baseline consistently in BLEU, NIST and METEOR.

There is a common phenomenon among SMP, SSP and ESSP. $w_{ch}pw_{en}$ always performs better than the other two cases. It seems that Chinese word prefers to have English pseudo-word equivalence which has more than or equal to one word. $pw_{ch}pw_{en}$ in ESSP performs similar to the baseline, which reflects that our direct pseudo-word pairs do not work very well with GIZA++ alignments. Such disagreement is weakened by using pseudo-words on only one language side ($w_{ch}pw_{en}$ or $pw_{ch}w_{en}$), while the advantage of pseudo-words is still leveraged in the alignments.

Best ESSP ($w_{ch}pw_{en}$) is significantly better than baseline ($p<0.01$) in BLEU score, best SMP ($w_{ch}pw_{en}$) is significantly better than baseline ($p<0.05$) in BLEU score. This indicates that pseudo-words, through either monolingual searching or synchronous searching, are more effective than words as to being basic translational units.

Figure 5 illustrates examples of pseudo-words of one Chinese-to-English sentence pair. Gold standard word alignments are shown at the bottom of figure 5. We can see that "front desk" is recognized as one pseudo-word in ESSP. Because SMP performs monolingually, it can not consider "前台" and "front desk" simultaneously. SMP only detects frequent monolingual multi-words as pseudo-words. SSP has a strong constraint that all parts of a sentence pair should be aligned, so source sentence and target sentence have same length after merging words into

	SMP			SSP			ESSP			baseline
	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	
BLEU	0.3996	0.4155	0.4024	0.3184	0.3661	0.3552	0.3998	**0.4229**	0.4147	0.4029
NIST	7.4711	**7.6452**	7.6186	6.4099	6.9284	6.8012	7.1665	7.4373	7.4235	7.0419
METEOR	0.5900	**0.6008**	0.6000	0.5255	0.5569	0.5454	0.5739	0.5963	0.5891	0.5785

Table 3. Performance of using pseudo-words on small data.

pseudo-words. We can see that too many pseudo-words are detected by SSP.

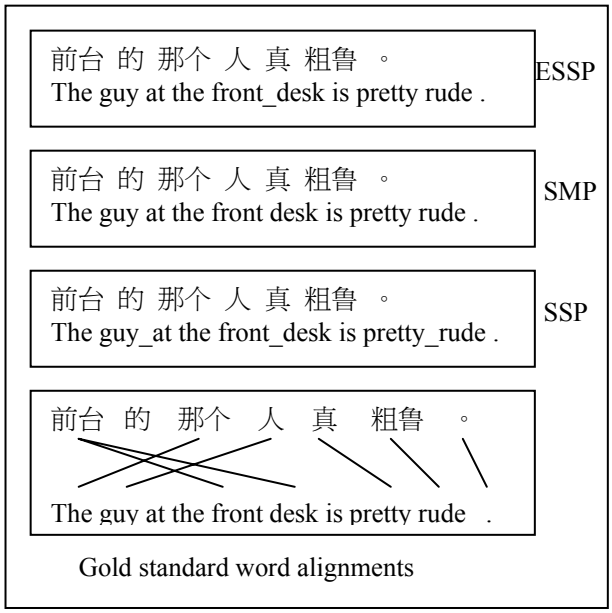


Figure 5. Outputs of the three algorithms ESSP, SMP and SSP on one sentence pair and gold standard word alignments. Words in one pseudo-word are concatenated by "_".

3.3.1 Pseudo-word Unpacking Performances on Small Corpus

We test pseudo-word unpacking in ESSP. Table 4 presents its performances on small corpus.

	$\text{unpacking}_{\text{ESSP}}$			baseline
	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	
BLEU	0.4097	**0.4182**	0.4031	0.4029
NIST	**7.5547**	7.2893	7.2670	7.0419
METEOR	**0.5951**	0.5874	0.5846	0.5785

Table 4. Performances of pseudo-word unpacking on small corpus.

We can see that pseudo-word unpacking significantly outperforms baseline. $w_{ch}pw_{en}$ is significantly better than baseline ($p<0.04$) in BLEU score. Unpacked pseudo-word performs comparatively with pseudo-word without unpacking. There is no statistical difference between them. It shows that the improvement derives from pseudo-word itself as basic translational unit, does not rely very much on higher language model order or longer max phrase length setting.

3.4 Pseudo-word Performances on Large Corpus

Table 5 lists the performance of using pseudo-words on large corpus. We apply SMP on this task. ESSP is not applied because of its high computational complexity. Table 5 shows that all three configurations ($pw_{ch}pw_{en}$, $w_{ch}pw_{en}$, $pw_{ch}w_{en}$) of SMP outperform the baseline. If we go back to the definition of sequence significance, we can see that it is a data-driven definition that utilizes corpus frequencies. Corpus scale has an influence on computation of sequence significance in long sentences which appear frequently in news domain. SMP benefits from large corpus, and $w_{ch}pw_{en}$ is significantly better than baseline ($p<0.01$). Similar to performances on small corpus, $w_{ch}pw_{en}$ always performs better than the other two cases, which indicates that Chinese word prefers to have English pseudo-word equivalence which has more than or equal to one word.

	SMP			baseline
	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	
BLEU	0.3185	**0.3230**	0.3166	0.3146
NIST	8.9216	**9.0447**	8.9210	8.8462
METEOR	0.5402	**0.5489**	0.5435	0.5335

Table 5. Performance of using pseudo-words on large corpus.

3.4.1 Pseudo-word Unpacking Performances on Large Corpus

Table 6 presents pseudo-word unpacking performances on large corpus. All three configurations improve performance over baseline after pseudo-word unpacking. $pw_{ch}pw_{en}$ attains the best BLEU among the three configurations, and is significantly better than baseline ($p<0.03$). $w_{ch}pw_{en}$ is also significantly better than baseline ($p<0.04$). By comparing table 6 with table 5, we can see that unpacked pseudo-word performs comparatively with pseudo-word without unpacking. There is no statistical difference be-

tween them. It shows that the improvement derives from pseudo-word itself as basic translational unit, does not rely very much on higher language model order or longer max phrase length setting. In fact, slight improvement in $pw_{ch}pw_{en}$ and $pw_{ch}w_{en}$ is seen after pseudo-word unpacking, which indicates that higher language model order and longer max phrase length impact the performance in these two configurations.

	Unpacking$_{SMP}$			Baseline
	$pw_{ch}pw_{en}$	$w_{ch}pw_{en}$	$pw_{ch}w_{en}$	
BLEU	**0.3219**	0.3192	0.3187	0.3146
NIST	8.9458	8.9325	**8.9801**	8.8462
METEOR	**0.5429**	0.5424	0.5411	0.5335

Table 6. Performance of pseudo-word unpacking on large corpus.

3.5 Comparison to English Chunking

English chunking is experimented to compare with pseudo-word. We use FlexCRFs (Xuan-Hieu Phan et al., 2005) to get English chunks. Since there is no standard Chinese chunking data and code, only English chunking is executed. The experimental results show that English chunking performs far below baseline, usually 8 absolute BLEU points below. It shows that simple chunks are not suitable for being basic translational units.

4 Conclusion

We have presented pseudo-word as a novel machine translational unit for phrase-based machine translation. It is proposed to replace too fine-grained word as basic translational unit. Pseudo-word is a kind of basic multi-word expression that characterizes minimal sequence of consecutive words in sense of translation. By casting pseudo-word searching problem into a parsing framework, we search for pseudo-words in polynomial time. Experimental results of Chinese-to-English translation task show that, in phrase-based machine translation model, pseudo-word performs significantly better than word in both spoken language translation domain and news domain. Removing the power of higher order language model and longer max phrase length, which are inherent in pseudo-words, shows that pseudo-words still improve translational performance significantly over unary words.

References

S. Banerjee, and A. Lavie. 2005. *METEOR: An automatic metric for MT evaluation with improved correlation with human judgments*. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (ACL'05). 65–72.

P. Blunsom, T. Cohn, C. Dyer, M. Osborne. 2009. *A Gibbs Sampler for Phrasal Synchronous Grammar Induction.* In Proceedings of ACL-IJCNLP, Singapore.

P. Blunsom, T. Cohn, M. Osborne. 2008. *Bayesian synchronous grammar induction.* In Proceedings of NIPS 21, Vancouver, Canada.

P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. 1993. *The mathematics of machine translation: Parameter estimation.* Computational Linguistics, 19:263–312.

P.-C. Chang, M. Galley, and C. D. Manning. 2008. *Optimizing Chinese word segmentation for machine translation performance.* In Proceedings of the 3rd Workshop on Statistical Machine Translation (SMT'08). 224–232.

Chen, Stanley F. and Joshua Goodman. 1998. *An empirical study of smoothing techniques for language modeling.* Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

C. Cherry, D. Lin. 2007. *Inversion transduction grammar for joint phrasal translation modeling.* In Proc. of the HLTNAACL Workshop on Syntax and Structure in Statistical Translation (SSST 2007), Rochester, USA.

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

Y. Deng and W. Byrne. 2005. *HMM word and phrase alignment for statistical machine translation.* In Proc. of HLT-EMNLP, pages 169–176.

G. Doddington. 2002. *Automatic evaluation of machine translation quality using n-gram cooccurrence statistics.* In Proceedings of the 2nd International Conference on Human Language Technology (HLT'02). 138–145.

Kneser, Reinhard and Hermann Ney. 1995. *Improved backing-off for M-gram language modeling.* In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pages 181−184, Detroit, MI.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan,W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst. 2007. *Moses: Open source toolkit for statistical machine translation.* In Proc. of the

45th Annual Meeting of the ACL (ACL-2007), Prague.

P. Koehn, F. J. Och, D. Marcu. 2003. *Statistical phrasebased translation.* In Proc. of the 3rd International conference on Human Language Technology Research and 4th Annual Meeting of the NAACL (HLT-NAACL 2003), 81–88, Edmonton, Canada.

P. Koehn. 2004. *Statistical Significance Tests for Machine Translation Evaluation.* In Proceedings of EMNLP.

P. Lambert and R. Banchs. 2005. *Data Inferred Multi-word Expressions for Statistical Machine Translation.* In Proceedings of MT Summit X.

Y. Ma, N. Stroppa, and A. Way. 2007. *Bootstrapping word alignment via word packing.* In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07). 304–311.

Y. Ma, and A. Way. 2009. *Bilingually Motivated Word Segmentation for Statistical Machine Translation.* In ACM Transactions on Asian Language Information Processing, 8(2).

D. Marcu,W.Wong. 2002. *A phrase-based, joint probability model for statistical machine translation.* In Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002), 133–139, Philadelphia. Association for Computational Linguistics.

F. J. Och. 2003. *Minimum error rate training in statistical machine translation.* In Proc. of ACL, pages 160–167.

F. J. Och and H. Ney. 2003. *A systematic comparison of various statistical alignment models.* Computational Linguistics, 29(1):19–51.

Xuan-Hieu Phan, Le-Minh Nguyen, and Cam-Tu Nguyen. 2005. *FlexCRFs: Flexible Conditional Random Field Toolkit,* http://flexcrfs.sourceforge.net

K. Papineni, S. Roukos, T. Ward, W. Zhu. 2001. *Bleu: a method for automatic evaluation of machine translation,* 2001.

M. Paul, 2008. *Overview of the IWSLT 2008 evaluation campaign.* In Proc. of Internationa Workshop on Spoken Language Translation, 20-21 October 2008.

A. Stolcke. (2002). *SRILM - an extensible language modeling toolkit.* In Proceedings of ICSLP, Denver, Colorado.

D. Wu. 1997. *Stochastic inversion transduction grammars and bilingual parsing of parallel corpora.* Computational Linguistics, 23(3):377–403.

J. Xu, Zens., and H. Ney. 2004. *Do we need Chinese word segmentation for statistical machine translation?* In Proceedings of the ACL Workshop on Chinese Language Processing SIGHAN'04). 122–128.

J. Xu, J. Gao, K. Toutanova, and H. Ney. 2008. *Bayesian semi-supervised chinese word segmentation for statistical machine translation.* In Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08). 1017–1024.

H. Zhang, C. Quirk, R. C. Moore, D. Gildea. 2008. *Bayesian learning of non-compositional phrases with synchronous parsing.* In Proc. of the 46th Annual Conference of the Association for Computational Linguistics: Human Language Technologies (ACL-08:HLT), 97–105, Columbus, Ohio.

R. Zhang, K. Yasuda, and E. Sumita. 2008. *Improved statistical machine translation by multiple Chinese word segmentation.* In Proceedings of the 3rd Workshop on Statistical Machine Translation (SMT'08). 216–223.

Hierarchical Search for Word Alignment

Jason Riesa and **Daniel Marcu**
Information Sciences Institute
Viterbi School of Engineering
University of Southern California
{riesa, marcu}@isi.edu

Abstract

We present a simple yet powerful hierarchical search algorithm for automatic word alignment. Our algorithm induces a forest of alignments from which we can efficiently extract a ranked *k*-best list. We score a given alignment within the forest with a flexible, linear discriminative model incorporating hundreds of features, and trained on a relatively small amount of annotated data. We report results on Arabic-English word alignment and translation tasks. Our model outperforms a GIZA++ Model-4 baseline by 6.3 points in F-measure, yielding a 1.1 BLEU score increase over a state-of-the-art syntax-based machine translation system.

1 Introduction

Automatic word alignment is generally accepted as a first step in training any statistical machine translation system. It is a vital prerequisite for generating translation tables, phrase tables, or syntactic transformation rules. Generative alignment models like IBM Model-4 (Brown et al., 1993) have been in wide use for over 15 years, and while not perfect (see Figure 1), they are completely unsupervised, requiring no annotated training data to learn alignments that have powered many current state-of-the-art translation system.

Today, there exist human-annotated alignments and an abundance of other information for many language pairs potentially useful for inducing accurate alignments. How can we take advantage of all of this data at our fingertips? Using feature functions that encode extra information is one good way. Unfortunately, as Moore (2005) points out, it is usually difficult to extend a given generative model with feature functions without changing the entire generative story. This difficulty has motivated much recent work in discriminative modeling for word alignment (Moore, 2005; Ittycheriah and Roukos, 2005; Liu et al., 2005; Taskar et al., 2005; Blunsom and Cohn, 2006; Lacoste-Julien et al., 2006; Moore et al., 2006).

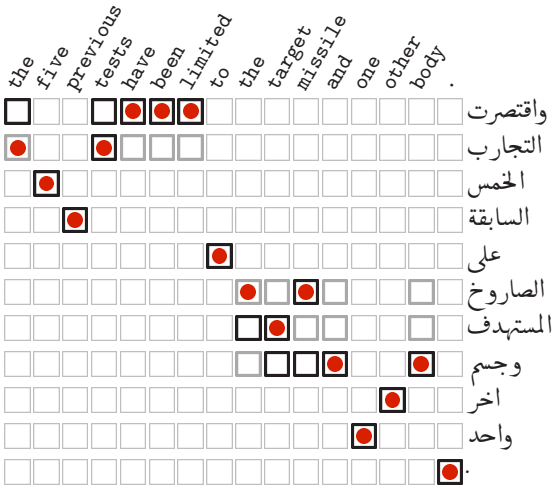


Figure 1: Model-4 alignment vs. a gold standard. Circles represent links in a human-annotated alignment, and black boxes represent links in the Model-4 alignment. Bold gray boxes show links gained after fully connecting the alignment.

We present in this paper a discriminative alignment model trained on relatively little data, with a simple, yet powerful hierarchical search procedure. We borrow ideas from both *k*-best parsing (Klein and Manning, 2001; Huang and Chiang, 2005; Huang, 2008) and forest-based, and hierarchical phrase-based translation (Huang and Chiang, 2007; Chiang, 2007), and apply them to word alignment.

Using a foreign string and an English parse tree as input, we formulate a bottom-up search on the parse tree, with the structure of the tree as a backbone for building a hypergraph of possible alignments. Our algorithm yields a forest of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 157–166,
Uppsala, Sweden, 11-16 July 2010.

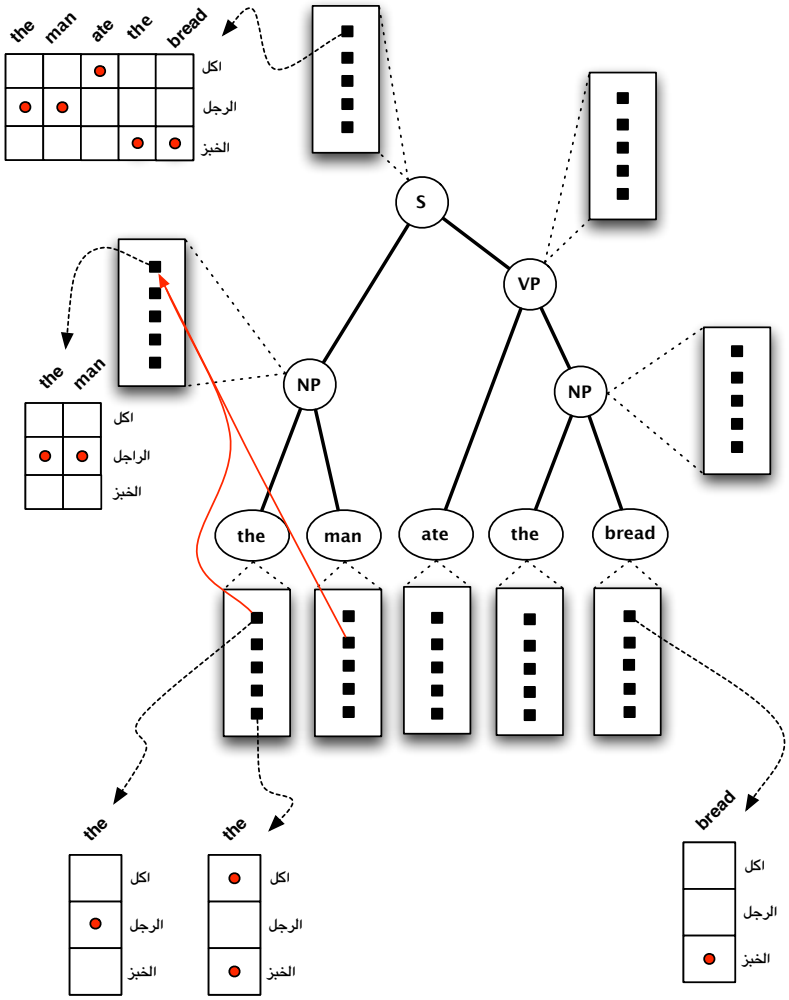


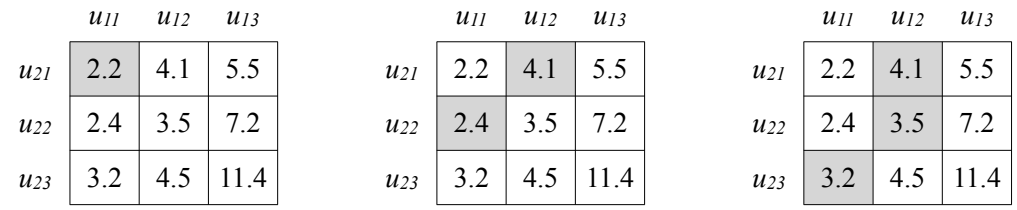
Figure 2: Example of approximate search through a hypergraph with beam size = 5. Each black square implies a partial alignment. Each partial alignment at each node is ranked according to its model score. In this figure, we see that the partial alignment implied by the 1-best hypothesis at the leftmost NP node is constructed by composing the best hypothesis at the terminal node labeled "the" and the 2nd-best hypothesis at the terminal node labeled "man". (We ignore terminal nodes in this toy example.) Hypotheses at the root node imply full alignment structures.

word alignments, from which we can efficiently extract the k-best. We handle an arbitrary number of features, compute them efficiently, and score alignments using a linear model. We train the parameters of the model using averaged perceptron (Collins, 2002) modified for structured outputs, but can easily fit into a max-margin or related framework. Finally, we use relatively little training data to achieve accurate word alignments. Our model can generate arbitrary alignments and learn from arbitrary gold alignments.

2 Word Alignment as a Hypergraph

Algorithm input The input to our alignment algorithm is a sentence-pair (e_1^n, f_1^m) and a parse tree over one of the input sentences. In this work, we parse our English data, and for each sentence $E = e_1^n$, let T be its syntactic parse. To generate parse trees, we use the Berkeley parser (Petrov et al., 2006), and use Collins head rules (Collins, 2003) to head-out binarize each tree.

Overview We present a brief overview here and delve deeper in Section 2.1. Word alignments are built bottom-up on the parse tree. Each node v in the tree holds *partial alignments* sorted by score.



(a) Score the left corner alignment first. Assume it is the 1-best. Numbers in the rest of the boxes are hidden at this point.

(b) Expand the frontier of alignments. We are now looking for the 2nd best.

(c) Expand the frontier further. After this step we have our top k alignments.

Figure 3: Cube pruning with alignment hypotheses to select the top-k alignments at node v with children $\langle u_1, u_2 \rangle$. In this example, $k = 3$. Each box represents the combination of two partial alignments to create a larger one. The score in each box is the sum of the scores of the child alignments plus a combination cost.

Each partial alignment comprises the columns of the alignment matrix for the e-words spanned by v, and each is scored by a linear combination of feature functions. See Figure 2 for a small example.

Initial partial alignments are enumerated and scored at preterminal nodes, each spanning a single column of the word alignment matrix. To speed up search, we can prune at each node, keeping a beam of size k. In the diagram depicted in Figure 2, the beam is size $k = 5$.

From here, we traverse the tree nodes bottom-up, combining partial alignments from child nodes until we have constructed a single full alignment at the root node of the tree. If we are interested in the k-best, we continue to populate the root node until we have k alignments.[1]

We use one set of feature functions for preterminal nodes, and another set for nonterminal nodes. This is analogous to local and nonlocal feature functions for parse-reranking used by Huang (2008). Using nonlocal features at a nonterminal node emits a combination cost for composing a set of child partial alignments.

Because combination costs come into play, we use cube pruning (Chiang, 2007) to approximate the k-best combinations at some nonterminal node v. Inference is exact when only local features are used.

Assumptions There are certain assumptions related to our search algorithm that we must make: (1) that using the structure of 1-best English syntactic parse trees is a reasonable way to frame and drive our search, and (2) that F-measure approximately decomposes over hyperedges.

We perform an oracle experiment to validate these assumptions. We find the oracle for a given (T,e,f) triple by proceeding through our search algorithm, forcing ourselves to always select correct links with respect to the gold alignment when possible, breaking ties arbitrarily. The the F_1 score of our oracle alignment is 98.8%, given this "perfect" model.

2.1 Hierarchical search

Initial alignments We can construct a word alignment hierarchically, bottom-up, by making use of the structure inherent in syntactic parse trees. We can think of building a word alignment as filling in an $M \times N$ matrix (Figure 1), and we begin by visiting each preterminal node in the tree. Each of these nodes spans a single e word. (Line 2 in Algorithm 1).

From here we can assign links from each e word to zero or more f words (Lines 6–14). At this level of the tree the span size is 1, and the partial alignment we have made spans a single column of the matrix. We can make many such partial alignments depending on the links selected. Lines 5 through 9 of Algorithm 1 enumerate either the null alignment, single-link alignments, or two-link alignments. Each partial alignment is scored and stored in a sorted heap (Lines 9 and 13).

In practice enumerating all two-link alignments can be prohibitive for long sentence pairs; we set a practical limit and score only pairwise combina-

[1] We use approximate dynamic programming to store alignments, keeping only scored lists of pointers to initial single-column spans. Each item in the list is a derivation that implies a partial alignment.

Algorithm 1: Hypergraph Alignment

Input:
Source sentence e_1^n
Target sentence f_1^m
Parse tree T over e_1^n
Set of feature functions $\mathbf{h}$
Weight vector $\mathbf{w}$
Beam size k

Output:
A k-best list of alignments over e_1^n and f_1^m

1 **function** ALIGN(e_1^n, f_1^m, T)
2 **for** $v \in T$ *in bottom-up order* **do**
3 $\alpha_v \leftarrow \emptyset$
4 **if** IS-PRETERMINALNODE(v) **then**
5 $i \leftarrow$ index-of(v)
6 **for** $j = 0$ **to** m **do**
7 $links \leftarrow (i, j)$
8 $score \leftarrow \mathbf{w} \cdot \mathbf{h}(links, v, e_1^n, f_1^m)$
9 PUSH(α_v, $\langle score, links \rangle$, k)
10 **for** $k = j + 1$ **to** m **do**
11 $links \leftarrow (i, j), (i, k)$
12 $score \leftarrow \mathbf{w} \cdot \mathbf{h}(links, v, e_1^n, f_1^m)$
13 PUSH(α_v, $\langle score, links \rangle$, k)
14 **end**
15 **end**
16 **else**
17 $\alpha_v \leftarrow$ GROWSPAN(children(v), k)
18 **end**
19 **end**
20 **end**
21 **function** GROWSPAN($\langle u_1, u_2 \rangle, k$)
22 **return** CUBEPRUNING($\langle \alpha_{u_1}, \alpha_{u_2} \rangle, k$,**w**,**h**)
23 **end**

tions of the top $n = \max\left\{\frac{|f|}{2}, 10\right\}$ scoring single-link alignments.

We limit the number of total partial alignments α_v kept at each node to k. If at any time we wish to push onto the heap a new partial alignment when the heap is full, we pop the current worst off the heap and replace it with our new partial alignment if its score is better than the current worst.

Building the hypergraph We now visit internal nodes (Line 16) in the tree in bottom-up order. At each nonterminal node v we wish to combine the partial alignments of its children $u_1, \ldots, u_c$. We use cube pruning (Chiang, 2007; Huang and Chiang, 2007) to select the k-best combinations of the partial alignments of $u_1, \ldots, u_c$ (Line 19). Note

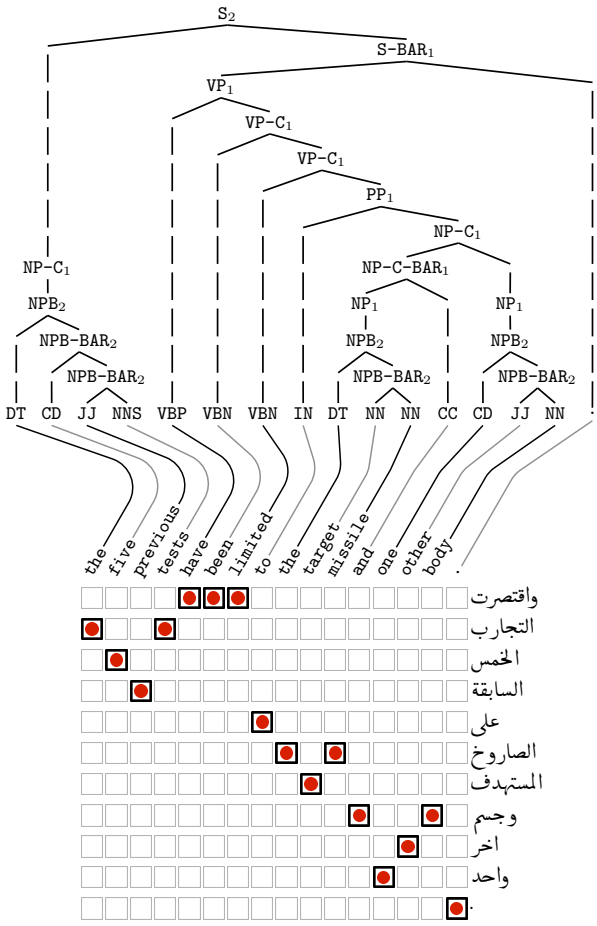


Figure 4: Correct version of Figure 1 after hypergraph alignment. Subscripts on the nonterminal labels denote the branch containing the head word for that span.

that Algorithm 1 assumes a binary tree[2], but is not necessary. In the general case, cube pruning will operate on a d-dimensional hypercube, where d is the branching factor of node v.

We cannot enumerate and score every possibility; without the cube pruning approximation, we will have k^c possible combinations at each node, exploding the search space exponentially. Figure 3 depicts how we select the top-k alignments at a node v from its children $\langle u_1, u_2 \rangle$.

3 Discriminative training

We incorporate all our new features into a linear model and learn weights for each using the online averaged perceptron algorithm (Collins, 2002) with a few modifications for structured outputs inspired by Chiang et al. (2008). We define:

[2]We find empirically that using binarized trees reduces search errors in cube pruning.

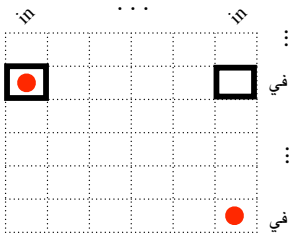


Figure 5: A common problem with GIZA++ Model 4 alignments is a weak distortion model. The second English "in" is aligned to the wrong Arabic token. Circles show the gold alignment.

$$\gamma(y) = \ell(y_i, y) + \mathbf{w} \cdot (\mathbf{h}(y_i) - \mathbf{h}(y)) \quad (1)$$

where $\ell(y_i,y)$ is a loss function describing how bad it is to guess y when the correct answer is y_i. In our case, we define $\ell(y_i,y)$ as $1-F_1(y_i,y)$. We select the oracle alignment according to:

$$y^+ = \arg\min_{y \in \text{CAND}(x)} \gamma(y) \quad (2)$$

where $\text{CAND}(x)$ is a set of hypothesis alignments generated from input x. Instead of the traditional oracle, which is calculated solely with respect to the loss $\ell(y_i,y)$, we choose the oracle that jointly minimizes the loss and the difference in model score to the true alignment. Note that Equation 2 is equivalent to maximizing the sum of the F-measure and model score of y:

$$y^+ = \arg\max_{y \in \text{CAND}(x)} (F_1(y_i, y) + \mathbf{w} \cdot \mathbf{h}(y)) \quad (3)$$

Let $\hat{y}$ be the 1-best alignment according to our model:

$$\hat{y} = \arg\max_{y \in \text{CAND}(x)} \mathbf{w} \cdot \mathbf{h}(y) \quad (4)$$

Then, at each iteration our weight update is:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(\mathbf{h}(y^+) - \mathbf{h}(\hat{y})) \quad (5)$$

where η is a learning rate parameter.[3] We find that this more conservative update gives rise to a much more stable search. After each iteration, we expect y^+ to get closer and closer to the true y_i.

4 Features

Our simple, flexible linear model makes it easy to throw in many features, mapping a given complex alignment structure into a single high-dimensional feature vector. Our hierarchical search framework allows us to compute these features when needed, and affords us extra useful syntactic information.

We use two classes of features: *local* and *non-local*. Huang (2008) defines a feature h to be local if and only if it can be factored among the local productions in a tree, and non-local otherwise. Analogously for alignments, our class of local features are those that can be factored among the local partial alignments competing to comprise a larger span of the matrix, and non-local otherwise. These features score a set of links and the words connected by them.

Feature development Our features are inspired by analysis of patterns contained among our gold alignment data and automatically generated parse trees. We use both local lexical and nonlocal structural features as described below.

4.1 Local features

These features fire on single-column spans.

- From the output of GIZA++ Model 4, we compute **lexical probabilities** $p(e \mid f)$ and $p(f \mid e)$, as well as a **fertility table** $\phi(e)$. From the fertility table, we fire features $\phi_0(e)$, $\phi_1(e)$, and $\phi_{2+}(e)$ when a word e is aligned to zero, one, or two or more words, respectively. Lexical probability features $p(e \mid f)$ and $p(f \mid e)$ fire when a word e is aligned to a word f.

- Based on these features, we include a binary **lexical-zero** feature that fires if both $p(e \mid f)$ and $p(f \mid e)$ are equal to zero for a given word pair (e, f). Negative weights essentially penalize alignments with links never seen before in the Model 4 alignment, and positive weights encourage such links. We employ a separate instance of this feature for each English part-of-speech tag: $p(f \mid e, t)$.

  We learn a different feature weight for each. Critically, this feature tells us how much to trust alignments involving nouns, verbs, adjectives, function words, punctuation, etc. from the Model 4 alignments from which our $p(e \mid f)$ and $p(f \mid e)$ tables are built. Table 1 shows a sample of learned weights. Intuitively, alignments involving English parts-of-speech more likely to be content words (e.g. NNPS, NNS, NN) are more trustworthy

[3] We set η to 0.05 in our experiments.

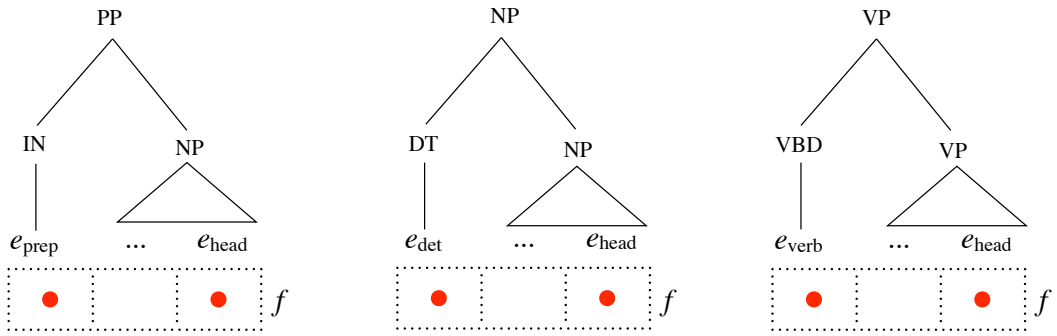


Figure 6: Features PP-NP-head, NP-DT-head, and VP-VP-head fire on these tree-alignment patterns. For example, PP-NP-head fires exactly when the head of the PP is aligned to exactly the same f words as the head of it's sister NP.

	Penalty
NNPS	−1.11
NNS	−1.03
NN	−0.80
NNP	−0.62
VB	−0.54
VBG	−0.52
JJ	−0.50
JJS	−0.46
VBN	−0.45
⋮	⋮
POS	−0.0093
EX	−0.0056
RP	−0.0037
WP$	−0.0011
TO	0.037
	Reward

Table 1: A sampling of learned weights for the **lexical zero** feature. Negative weights penalize links never seen before in a baseline alignment used to initialize lexical $p(e \mid f)$ and $p(f \mid e)$ tables. Positive weights outright reward such links.

than those likely to be function words (e.g. TO, RP, EX), where the use of such words is often radically different across languages.

- We also include a measure of **distortion**. This feature returns the distance to the diagonal of the matrix for any link in a partial alignment. If there is more than one link, we return the distance of the link farthest from the diagonal.

- As a lexical backoff, we include a **tag probability** feature, $p(t \mid f)$ that fires for some link (e, f) if the part-of-speech tag of e is t. The conditional probabilities in this table are computed from our parse trees and the baseline Model 4 alignments.

- In cases where the lexical probabilities are too strong for the distortion feature to overcome (see Figure 5), we develop the **multiple-distortion** feature. Although local features do not know the partial alignments at other spans, they do have access to the entire English sentence at every step because our input is constant. If some e exists more than once in e_1^n we fire this feature on all links containing word e, returning again the distance to the diagonal for that link. We learn a strong negative weight for this feature.

- We find that binary **identity** and **punctuation-mismatch** features are important. The binary identity feature fires if $e = f$, and proves useful for untranslated numbers, symbols, names, and punctuation in the data. Punctuation-mismatch fires on any link that causes nonpunctuation to be aligned to punctuation.

Additionally, we include fine-grained versions of the lexical probability, fertility, and distortion features. These fire for for each link (e, f) and part-of-speech tag. That is, we learn a separate weight for each feature for each part-of-speech tag in our data. Given the tag of e, this affords the model the ability to pay more or less attention to the features described above depending on the tag given to e.

Arabic-English specific features We describe here language specific features we implement to exploit shallow Arabic morphology.

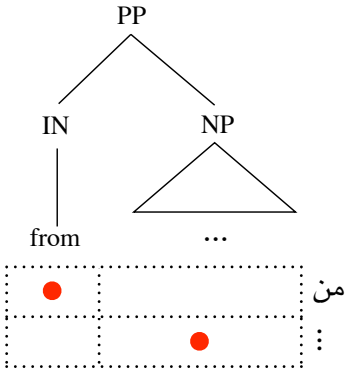


Figure 7: This figure depicts the tree/alignment structure for which the feature **PP-from-prep** fires. The English preposition "from" is aligned to Arabic word من. Any aligned words in the span of the sister NP are aligned to words following من. English preposition structure commonly matches that of Arabic in our gold data. This family of features captures these observations.

- We observe the Arabic prefix و, transliterated `w-` and generally meaning *and*, to prepend to most any word in the lexicon, so we define features $p_{\neg w}(e \mid f)$ and $p_{\neg w}(f \mid e)$. If f begins with `w-`, we strip off the prefix and return the values of $p(e \mid f)$ and $p(f \mid e)$. Otherwise, these features return 0.
- We also include analogous feature functions for several functional and pronominal prefixes and suffixes.[4]

4.2 Nonlocal features

These features comprise the combination cost component of a partial alignment score and may fire when concatenating two partial alignments to create a larger span. Because these features can look into any two arbitrary subtrees, they are considered nonlocal features as defined by Huang (2008).

- Features **PP-NP-head**, **NP-DT-head**, and **VP-VP-head** (Figure 6) all exploit headwords on the parse tree. We observe English prepositions and determiners to often align to the headword of their sister. Likewise, we observe the head of a VP to align to the head of an immediate sister VP.

In Figure 4, when the search arrives at the left-most NPB node, the NP-DT-head feature will fire given this structure and links over the span `[the ... tests]`. When search arrives at the second NPB node, it will fire given the structure and links over the span `[the ... missle]`, but will not fire at the right-most NPB node.

- Local lexical preference features compete with the headword features described above. However, we also introduce nonlocal lexicalized features for the most common types of English and foreign prepositions to also compete with these general headword features.

  PP features **PP-of-prep**, **PP-from-prep**, **PP-to-prep**, **PP-on-prep**, and **PP-in-prep** fire at any PP whose left child is a preposition and right child is an NP. The head of the PP is one of the enumerated English prepositions and is aligned to any of the three most common foreign words to which it has also been observed aligned in the gold alignments. The last constraint on this pattern is that all words under the span of the sister NP, if aligned, must align to words following the foreign preposition. Figure 7 illustrates this pattern.

- Finally, we have a **tree-distance** feature to avoid making too many many-to-one (from many English words to a single foreign word) links. This is a simplified version of and similar in spirit to the tree distance metric used in (DeNero and Klein, 2007). For any pair of links (e_i, f) and (e_j, f) in which the e words differ but the f word is the same token in each, return the tree height of first common ancestor of e_i and e_j.

  This feature captures the intuition that it is much worse to align two English words at different ends of the tree to the same foreign word, than it is to align two English words under the same NP to the same foreign word.

  To see why a string distance feature that counts only the flat horizontal distance from e_i to e_j is not the best strategy, consider the following. We wish to align a determiner to the same f word as its sister head noun under the same NP. Now suppose there are several intermediate adjectives separating the determiner and noun. A string distance met-

[4] Affixes used by our model are currently: بـ, لـ, الـ, بالـ, ـي, ـكم, ـكما, ـهم, ـهما. Others either we did not experiment with, or seemed to provide no significant benefit, and are not included.

ric, with no knowledge of the relationship between determiner and noun will levy a much heavier penalty than its tree distance analog.

5 Related Work

Recent work has shown the potential for syntactic information encoded in various ways to support inference of superior word alignments. Very recent work in word alignment has also started to report downstream effects on BLEU score.

Cherry and Lin (2006) introduce soft syntactic ITG (Wu, 1997) constraints into a discriminative model, and use an ITG parser to constrain the search for a Viterbi alignment. Haghighi et al. (2009) confirm and extend these results, showing BLEU improvement for a hierarchical phrase-based MT system on a small Chinese corpus. As opposed to ITG, we use a linguistically motivated phrase-structure tree to drive our search and inform our model. And, unlike ITG-style approaches, our model can generate arbitrary alignments and learn from arbitrary gold alignments.

DeNero and Klein (2007) refine the distortion model of an HMM aligner to reflect tree distance instead of string distance. Fossum et al. (2008) start with the output from GIZA++ Model-4 union, and focus on increasing precision by deleting links based on a linear discriminative model exposed to syntactic and lexical information.

Fraser and Marcu (2007) take a semi-supervised approach to word alignment, using a small amount of gold data to further tune parameters of a headword-aware generative model. They show a significant improvement over a Model-4 union baseline on a very large corpus.

6 Experiments

We evaluate our model and and resulting alignments on Arabic-English data against those induced by IBM Model-4 using GIZA++ (Och and Ney, 2003) with both the union and grow-diag-final heuristics. We use 1,000 sentence pairs and gold alignments from LDC2006E86 to train model parameters: 800 sentences for training, 100 for testing, and 100 as a second held-out development set to decide when to stop perceptron training. We also align the test data using GIZA++[5] along with 50 million words of English.

[5]We use a standard training procedure: 5 iterations of Model-1, 5 iterations of HMM, 3 iterations of Model-3, and 3 iterations of Model-4.

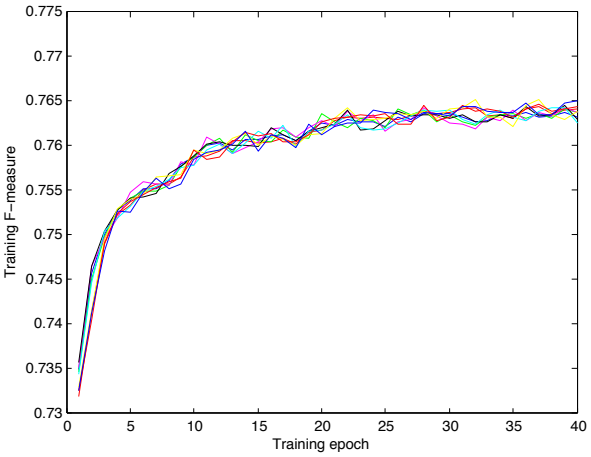


Figure 8: Learning curves for 10 random restarts over time for parallel averaged perceptron training. These plots show the current F-measure on the training set as time passes. Perceptron training here is quite stable, converging to the same general neighborhood each time.

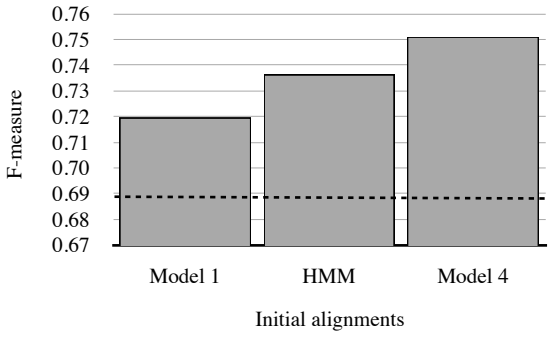


Figure 9: Model robustness to the initial alignments from which the $p(e \mid f)$ and $p(f \mid e)$ features are derived. The dotted line indicates the baseline accuracy of GIZA++ Model 4 alone.

6.1 Alignment Quality

We empirically choose our beam size k from the results of a series of experiments, setting k=1, 2, 4, 8, 16, 32, and 64. We find setting $k = 16$ to yield the highest accuracy on our held-out test data. Using wider beams results in higher F-measure on training data, but those gains do not translate into higher accuracy on held-out data.

The first three columns of Table 2 show the balanced F-measure, Precision, and Recall of our alignments versus the two GIZA++ Model-4 baselines. We report an F-measure 8.6 points over Model-4 union, and 6.3 points over Model-4 grow-diag-final.

	F	P	R	Arabic/English BLEU	# Unknown Words
M4 (union)	.665	.636	.696	45.1	2,538
M4 (grow-diag-final)	.688	.702	.674	46.4	2,262
Hypergraph alignment	**.751**	**.780**	**.724**	**47.5**	**1,610**

Table 2: F-measure, Precision, Recall, the resulting BLEU score, and number of unknown words on a held-out test corpus for three types of alignments. BLEU scores are case-insensitive IBM BLEU. We show a 1.1 BLEU increase over the strongest baseline, Model-4 grow-diag-final. This is statistically significant at the $p < 0.01$ level.

Figure 8 shows the stability of the search procedure over ten random restarts of parallel averaged perceptron training with 40 CPUs. Training examples are randomized at each epoch, leading to slight variations in learning curves over time but all converge into the same general neighborhood.

Figure 9 shows the robustness of the model to initial alignments used to derive lexical features $p(e \mid f)$ and $p(f \mid e)$. In addition to IBM Model 4, we experiment with alignments from Model 1 and the HMM model. In each case, we significantly outperform the baseline GIZA++ Model 4 alignments on a heldout test set.

6.2 MT Experiments

We align a corpus of 50 million words with GIZA++ Model-4, and extract translation rules from a 5.4 million word core subset. We align the same core subset with our trained hypergraph alignment model, and extract a second set of translation rules. For each set of translation rules, we train a machine translation system and decode a held-out test corpus for which we report results below.

We use a syntax-based translation system for these experiments. This system transforms Arabic strings into target English syntax trees Translation rules are extracted from (e-tree, f-string, alignment) triples as in (Galley et al., 2004; Galley et al., 2006).

We use a randomized language model (similar to that of Talbot and Brants (2008)) of 472 million English words. We tune the the parameters of the MT system on a held-out development corpus of 1,172 parallel sentences, and test on a held-out parallel corpus of 746 parallel sentences. Both corpora are drawn from the NIST 2004 and 2006 evaluation data, with no overlap at the document or segment level with our training data.

Columns 4 and 5 in Table 2 show the results of our MT experiments. Our hypergraph alignment algorithm allows us a 1.1 BLEU increase over the best baseline system, Model-4 grow-diag-final. This is statistically significant at the $p < 0.01$ level. We also report a 2.4 BLEU increase over a system trained with alignments from Model-4 union.

7 Conclusion

We have opened up the word alignment task to advances in hypergraph algorithms currently used in parsing and machine translation decoding. We treat word alignment as a parsing problem, and by taking advantage of English syntax and the hypergraph structure of our search algorithm, we report significant increases in both F-measure and BLEU score over standard baselines in use by most state-of-the-art MT systems today.

Acknowledgements

We would like to thank our colleagues in the Natural Language Group at ISI for many meaningful discussions and the anonymous reviewers for their thoughtful suggestions. This research was supported by DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies, and a USC CREATE Fellowship to the first author.

References

Phil Blunsom and Trevor Cohn. 2006. Discriminative Word Alignment with Conditional Random Fields. In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312. MIT Press. Camrbidge, MA. USA.

Colin Cherry and Dekang Lin. 2006. Soft Syntactic Constraints for Word Alignment through Discriminative Training. In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

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

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of EMNLP*. Honolulu, HI. USA.

Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*. 29(4):589–637. MIT Press. Cambridge, MA. USA.

Michael Collins 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL*. Prague, Czech Republic.

Alexander Fraser and Daniel Marcu. 2007. Getting the Structure Right for Word Alignment: LEAF. In *Proceedings of EMNLP-CoNLL*. Prague, Czech Republic.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using Syntax to Improve Word Alignment Precision for Syntax-Based Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Columbus, Ohio.

Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the 7th International Workshop on Parsing Technologies*. Beijing, China.

Aria Haghighi, John Blitzer, and Dan Klein. 2009. Better Word Alignments with Supervised ITG Models. In *Proceedings of ACL-IJCNLP 2009*. Singapore.

Liang Huang and David Chiang. 2005. Better *k*-best Parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*. Vancouver, BC. Canada.

Liang Huang and David Chiang. 2007. Forest Rescoring: Faster Decoding with Integrated Language Models. In *Proceedings of the 45th Annual Meeting of the ACL*. Prague, Czech Republic.

Liang Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *Proceedings of the 46th Annual Meeting of the ACL*. Columbus, OH. USA.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a Translation Rule? In *Proceedings of NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable Inference and Training of Context-Rich Syntactic Models In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*. Vancouver, BC. Canada.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. 2006. Word alignment via Quadratic Assignment. In *Proceedings of HLT-EMNLP*. New York, NY. USA.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear Models for Word Alignment In *Proceedings of the 43rd Annual Meeting of the ACL*. Ann Arbor, Michigan. USA.

Robert C. Moore. 2005. A Discriminative Framework for Word Alignment. In *Proceedings of EMNLP*. Vancouver, BC. Canada.

Robert C. Moore, Wen-tau Yih, and Andreas Bode. 2006. Improved Discriminative Bilingual Word Alignment In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*. 29(1):19–52. MIT Press. Cambridge, MA. USA.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein 2006. Learning Accurate, Compact, and Interpretable Tree Annotation In *Proceedings of the 44th Annual Meeting of the ACL*. Sydney, Australia.

Kishore Papineni, Salim Roukos, T. Ward, and W-J. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation In *Proceedings of the 40th Annual Meeting of the ACL*. Philadelphia, PA. USA.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. In *Proceedings of HLT-EMNLP*. Vancouver, BC. Canada.

David Talbot and Thorsten Brants. 2008. Randomized Language Models via Perfect Hash Functions. In *Proceedings of ACL-08: HLT*. Columbus, OH. USA.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*. 23(3):377–404. MIT Press. Cambridge, MA. USA.

"Was it good? It was provocative." Learning the meaning of scalar adjectives

Marie-Catherine de Marneffe, Christopher D. Manning and **Christopher Potts**
Linguistics Department
Stanford University
Stanford, CA 94305
{mcdm,manning,cgpotts}@stanford.edu

Abstract

Texts and dialogues often express information indirectly. For instance, speakers' answers to yes/no questions do not always straightforwardly convey a 'yes' or 'no' answer. The intended reply is clear in some cases (*Was it good? It was great!*) but uncertain in others (*Was it acceptable? It was unprecedented.*). In this paper, we present methods for interpreting the answers to questions like these which involve scalar modifiers. We show how to ground scalar modifier meaning based on data collected from the Web. We learn scales between modifiers and infer the extent to which a given answer conveys 'yes' or 'no'. To evaluate the methods, we collected examples of question–answer pairs involving scalar modifiers from CNN transcripts and the Dialog Act corpus and use response distributions from Mechanical Turk workers to assess the degree to which each answer conveys 'yes' or 'no'. Our experimental results closely match the Turkers' response data, demonstrating that meanings can be learned from Web data and that such meanings can drive pragmatic inference.

1 Introduction

An important challenge for natural language processing is how to learn not only basic linguistic meanings but also how those meanings are systematically enriched when expressed in context. For instance, answers to polar (yes/no) questions do not always explicitly contain a 'yes' or 'no', but rather give information that the hearer can use to infer such an answer in a context with some degree of certainty. Hockey et al. (1997) find that 27% of answers to polar questions do not contain a direct 'yes' or 'no' word, 44% of which they regard as failing to convey a clear 'yes' or 'no' response. In some cases, interpreting the answer is straightforward (*Was it bad? It was terrible.*), but in others, what to infer from the answer is unclear (*Was it good? It was provocative.*). It is even common for the speaker to explicitly convey his own uncertainty with such answers.

In this paper, we focus on the interpretation of answers to a particular class of polar questions: ones in which the main predication involves a gradable modifier (e.g., *highly unusual*, *not good*, *little*) and the answer either involves another gradable modifier or a numerical expression (e.g., *seven years old*, *twenty acres of land*). Interpreting such question–answer pairs requires dealing with modifier meanings, specifically, learning context-dependent scales of expressions (Horn, 1972; Fauconnier, 1975) that determine how, and to what extent, the answer as a whole resolves the issue raised by the question.

We propose two methods for learning the knowledge necessary for interpreting indirect answers to questions involving gradable adjectives, depending on the type of predications in the question and the answer. The first technique deals with pairs of modifiers: we hypothesized that online, informal review corpora in which people's comments have associated ratings would provide a general-purpose database for mining scales between modifiers. We thus use a large collection of online reviews to learn orderings between adjectives based on contextual entailment (*good* < *excellent*), and employ this scalar relationship to infer a yes/no answer (subject to negation and other qualifiers). The second strategy targets numerical answers. Since it is unclear what kind of corpora would contain the relevant information, we turn to the Web in general: we use distributional information retrieved via Web searches to assess whether the numerical measure counts as a posi-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 167–176,
Uppsala, Sweden, 11-16 July 2010.

tive or negative instance of the adjective in question. Both techniques exploit the same approach: using texts (the Web) to learn meanings that can drive pragmatic inference in dialogue. This paper demonstrates to some extent that meaning can be grounded from text in this way.

2 Related work

Indirect speech acts are studied by Clark (1979), Perrault and Allen (1980), Allen and Perrault (1980) and Asher and Lascarides (2003), who identify a wide range of factors that govern how speakers convey their intended messages and how hearers seek to uncover those messages from uncertain and conflicting signals. In the computational literature, Green and Carberry (1994, 1999) provide an extensive model that interprets and generates indirect answers to polar questions. They propose a logical inference model which makes use of discourse plans and coherence relations to infer *categorical* answers. However, to adequately interpret indirect answers, the uncertainty inherent in some answers needs to be captured (de Marneffe et al., 2009). While a straightforward 'yes' or 'no' response is clear in some indirect answers, such as in (1), the intended answer is less certain in other cases (2):[1]

(1) A: Do you think that's a good idea, that we just begin to ignore these numbers?
B: I think it's an excellent idea.

(2) A: Is he qualified?
B: I think he's young.

In (2), it might be that the answerer does not know about qualifications or does not want to talk about these directly, and therefore shifts the topic slightly. As proposed by Zeevat (1994) in his work on partial answers, the speaker's indirect answer might indicate that he is deliberately leaving the original question only partially addressed, while giving a fully resolving answer to another one. The hearer must then interpret the answer to work out the other question. In (2) in context, we get a sense that the speaker would resolve the issue to 'no', but that he is definitely not committed to that in any strong sense. Uncertainty can thus reside both on the speaker and the hearer sides, and the four following scenarios are attested in conversation:

a. The speaker is certain of 'yes' or 'no' and conveys that directly and successfully to the hearer.
b. The speaker is certain of 'yes' or 'no' but conveys this only partially to the hearer.
c. The speaker is uncertain of 'yes' or 'no' and conveys this uncertainty to the hearer.
d. The speaker is uncertain of 'yes' or 'no', but the hearer infers one of those with confidence.

The uncertainty is especially pressing for predications built around scalar modifiers, which are inherently vague and highly context-dependent (Kamp and Partee, 1995; Kennedy and McNally, 2005; Kennedy, 2007). For example, even if we fix the basic sense for *little* to mean 'young for a human', there is a substantial amount of gray area between the clear instances (babies) and the clear non-instances (adults). This is the source of uncertainty in (3), in which B's children fall into the gray area.

(3) A: Are your kids little?
B: I have a seven year-old and a ten year-old.

3 Corpus description

Since indirect answers are likely to arise in interviews, to gather instances of question–answer pairs involving gradable modifiers (which will serve to evaluate the learning techniques), we use online CNN interview transcripts from five different shows aired between 2000 and 2008 (Anderson Cooper, Larry King Live, Late Edition, Lou Dobbs Tonight, The Situation Room). We also searched the Switchboard Dialog Act corpus (Jurafsky et al., 1997). We used regular expressions and manual filtering to find examples of two-utterance dialogues in which the question and the reply contain some kind of gradable modifier.

3.1 Types of question–answer pairs

In total, we ended up with 224 question–answer pairs involving gradable adjectives. However our collection contains different types of answers, which naturally fall into two categories: (I) in 205 dialogues, both the question and the answer contain a gradable modifier; (II) in 19 dialogues, the reply contains a numerical measure (as in (3) above and (4)).

[1] Here and throughout, the examples come from the corpus described in section 3.

	Modification in answer	Example	Count
I	Other adjective	(1), (2)	125
	Adverb - same adjective	(5)	55
	Negation - same adjective	(6), (7)	21
	Omitted adjective	(8)	4
II	Numerical measure	(3), (4)	19

Table 1: Types of question–answer pairs, and counts in the corpus.

I	Modification in answer	Mean	SD
	Other adjective	1.1	0.6
	Adverb - same adjective	0.8	0.6
	Negation - same adjective	1.0	0.5
	Omitted adjective	1.1	0.2
II	Numerical measure	1.5	0.2

Table 2: Mean entropy values and standard deviation obtained in the Mechanical Turk experiment for each question–answer pair category.

(4) A: Have you been living there very long?
B: I'm in here right now about twelve and a half years.

Category I, which consists of pairs of modifiers, can be further divided. In most dialogues, the answer contains another adjective than the one used in the question, such as in (1). In others, the answer contains the same adjective as in the question, but modified by an adverb (e.g., *very*, *basically*, *quite*) as in (5) or a negation as in (6).

(5) A: That seems to be the biggest sign of progress there. Is that accurate?
B: That's absolutely accurate.

(6) A: Are you bitter?
B: I'm not bitter because I'm a soldier.

The negation can be present in the main clause when the adjectival predication is embedded, as in example (7).

(7) A: [. . .] Is that fair?
B: I don't think that's a fair statement.

In a few cases, when the question contains an adjective modifying a noun, the adjective is omitted in the answer:

(8) A: Is that a huge gap in the system?
B: It is a gap.

Table 1 gives the distribution of the types appearing in the corpus.

3.2 Answer assignment

To assess the degree to which each answer conveys 'yes' or 'no' in context, we use response distributions from Mechanical Turk workers. Given a written dialogue between speakers A and B, Turkers were asked to judge what B's answer conveys: 'definite yes', 'probable yes', 'uncertain', 'probable no', 'definite no'. Within each of the two 'yes' and 'no' pairs, there is a scalar relationship, but the pairs themselves are not in a scalar relationship with each other, and 'uncertain' is arguably a separate judgment. Figure 1 shows the exact formulation used in the experiment. For each dialogue, we got answers from 30 Turkers, and we took the dominant response as the correct one though we make extensive use of the full response distributions in evaluating our approach.[2] We also computed entropy values for the distribution of answers for each item. Overall, the agreement was good: 21 items have total agreement (entropy of 0.0 — 11 in the "adjective" category, 9 in the "adverb-adjective" category and 1 in the "negation" category), and 80 items are such that a single response got chosen 20 or more times (entropy < 0.9). The dialogues in (1) and (9) are examples of total agreement. In contrast, (10) has response entropy of 1.1, and item (11) has the highest entropy of 2.2.

(9) A: Advertisements can be good or bad. Was it a good ad?
B: It was a great ad.

(10) A: Am I clear?
B: I wish you were a little more forthright.

(11) A: 91 percent of the American people still express confidence in the long-term prospect of the U.S. economy; only 8 percent are not confident. Are they overly optimistic, in your professional assessment?

[2] 120 Turkers were involved (the median number of items done was 28 and the mean 56.5). The Fleiss' Kappa score for the five response categories is 0.46, though these categories are partially ordered. For the three-category response system used in section 5, which arguably has no scalar ordering, the Fleiss' Kappa is 0.63. Despite variant individual judgments, aggregate annotations done with Mechanical Turk have been shown to be reliable (Snow et al., 2008; Sheng et al., 2008; Munro et al., 2010). Here, the relatively low Kappa scores also reflect the uncertainty inherent in many of our examples, uncertainty that we seek to characterize and come to grips with computationally.

Indirect Answers to Yes/No Questions
In the following dialogue, speaker A asks a simple Yes/No question, but speaker B answers with something more indirect and complicated.

Which of the following best captures what speaker B meant here:

- B definitely meant to convey "Yes".
- B probably meant to convey "Yes".
- B definitely meant to convey "No".
- B probably meant to convey "No".
- (I really can't tell whether B meant to convey "Yes" or "No".)

Figure 1: Design of the Mechanical Turk experiment.

B: I think it shows how wise the American people are.

Table 2 shows the mean entropy values for the different categories identified in the corpus. Interestingly, the pairs involving an adverbial modification in the answer all received a positive answer ('yes' or 'probable yes') as dominant response. All 19 dialogues involving a numerical measure had either 'probable yes' or 'uncertain' as dominant response. There is thus a significant bias for positive answers: 70% of the category I items and 74% of the category II items have a positive answer as dominant response. Examining a subset of the Dialog Act corpus, we found that 38% of the yes/no questions receive a direct positive answers, whereas 21% have a direct negative answer. This bias probably stems from the fact that people are more likely to use an overt denial expression where they need to disagree, whether or not they are responding indirectly.

4 Methods

In this section, we present the methods we propose for grounding the meanings of scalar modifiers.

4.1 Learning modifier scales and inferring yes/no answers

The first technique targets items such as the ones in category I of our corpus. Our central hypothesis is that, in polar question dialogues, the semantic relationship between the main predication P_Q in the question and the main predication P_A in the answer is the primary factor in determining whether, and to what extent, 'yes' or 'no' was intended. If P_A is at least as strong as P_Q, the intended answer is 'yes'; if P_A is weaker than P_Q, the intended answer is 'no'; and, where no reliable entailment relationship exists between P_A and P_Q, the result is uncertainty.

For example, *good* is weaker (lower on the relevant scale) than *excellent*, and thus speakers infer that the reply in example (1) above is meant to convey 'yes'. In contrast, if we reverse the order of the modifiers — roughly, *Is it a great idea?*; *It's a good idea* — then speakers infer that the answer conveys 'no'. Had B replied with *It's a complicated idea* in (1), then uncertainty would likely have resulted, since *good* and *complicated* are not in a reliable scalar relationship. Negation reverses scales (Horn, 1972; Levinson, 2000), so it flips 'yes' and 'no' in these cases, leaving 'uncertain' unchanged. When both the question and the answer contain a modifier (such as in (9–11)), the yes/no response should correlate with the extent to which the pair of modifiers can be put into a scale based on contextual entailment.

To ground such scales from text, we collected a large corpus of online reviews from IMDB. Each of the reviews in this collection has an associated star rating: one star (most negative) to ten stars (most positive). Table 3 summarizes the distribution of reviews as well as the number of words and vocabulary across the ten rating categories.

As is evident from table 3, there is a significant bias for ten-star reviews. This is a common feature of such corpora of informal, user-provided reviews (Chevalier and Mayzlin, 2006; Hu et al., 2006; Pang and Lee, 2008). However, since we do not want to incorporate the linguistically uninteresting fact that people tend to write a lot of ten-star reviews, we assume uniform priors for the rating categories. Let $\text{count}(w, r)$ be the number of tokens of word w in reviews in rating category r, and let $\text{count}(r)$ be the total word count for all words in rating category r. The probability of w given a rating category r is simply $\Pr(w|r) = \text{count}(w, r)/\,\text{count}(r)$. Then under the assumption of uniform priors, we get $\Pr(r|w) = \Pr(w|r)/\sum_{r' \in R} \Pr(w|r')$.

In reasoning about our dialogues, we rescale the rating categories by subtracting 5.5 from each, to center them at 0. This yields the scale $R =$

Rating	Reviews	Words	Vocabulary	Average words per review
1	124,587	25,389,211	192,348	203.79
2	51,390	11,750,820	133,283	228.66
3	58,051	13,990,519	148,530	241.00
4	59,781	14,958,477	156,564	250.22
5	80,487	20,382,805	188,461	253.24
6	106,145	27,408,662	225,165	258.22
7	157,005	40,176,069	282,530	255.89
8	195,378	48,706,843	313,046	249.30
9	170,531	40,264,174	273,266	236.11
10	358,441	73,929,298	381,508	206.25
Total	1,361,796	316,956,878	1,160,072	206.25

Table 3: Numbers of reviews, words and vocabulary size per rating category in the IMDB review corpus, as well as the average number of words per review.

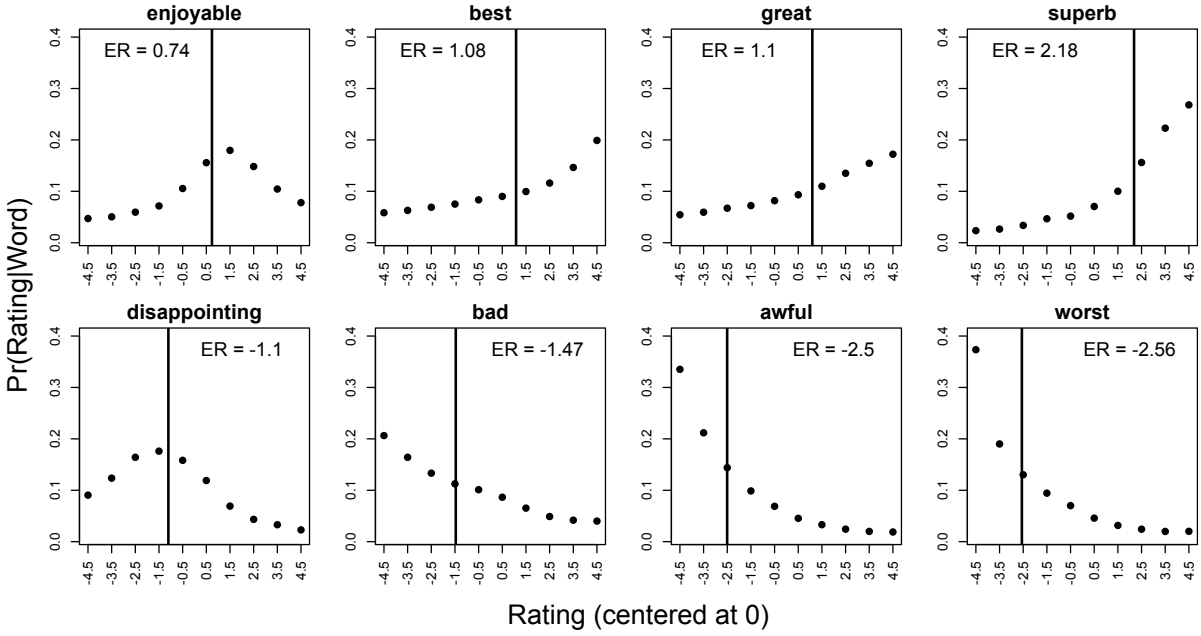


Figure 2: The distribution of some scalar modifiers across the ten rating categories. The vertical lines mark the expected ratings, defined as a weighted sum of the probability values (black dots).

$\langle -4.5, -3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.5, 4.5 \rangle$. Our rationale for this is that modifiers at the negative end of the scale (*bad*, *awful*, *terrible*) are not linguistically comparable to those at the positive end of the scale (*good*, *excellent*, *superb*). Each group forms its own qualitatively different scale (Kennedy and McNally, 2005). Rescaling allows us to make a basic positive vs. negative distinction. Once we have done that, an increase in absolute value is an increase in strength. In our experiments, we use expected rating values to characterize the polarity and strength of modifiers. The expected rating value for a word w is $\mathrm{ER}(w) = \sum_{r \in R} r \Pr(r|w)$. Figure 2 plots these values for a number of scalar terms, both positive and negative, across the rescaled ratings, with the vertical lines marking their ER values. The weak scalar modifiers all the way on the left are most common near the middle of the scale, with a slight positive bias in the top row and a slight negative bias in the bottom row. As we move from left to right, the bias for one end of the scale grows more extreme, until the words in question are almost never used outside of the most extreme rating category. The resulting scales correspond well with linguistic intuitions and thus provide an initial indication that the rating categories are a reliable guide to strength and polarity for scalar modifiers. We put this information to use in our dialogue corpus via the decision procedure

Let D be a dialogue consisting of (i) a polar question whose main predication is based on scalar predicate P_Q and (ii) an indirect answer whose main predication is based on scalar predicate P_A. Then:

1. if P_A or P_Q is missing from our data, infer 'Uncertain';
2. else if ER(P_Q) and ER(P_A) have different signs, infer 'No';
3. else if abs(ER(P_Q)) $\leqslant$ abs(ER(P_A)), infer 'Yes';
4. else infer 'No'.
5. In the presence of negation, map 'Yes' to 'No', 'No' to 'Yes', and 'Uncertain' to 'Uncertain'.

Figure 3: Decision procedure for using the word frequencies across rating categories in the review corpus to decide what a given answer conveys.

described in figure 3.

4.2 Interpreting numerical answers

The second technique aims at determining whether a numerical answer counts as a positive or negative instance of the adjective in the question (category II in our corpus).

Adjectives that can receive a conventional unit of measure, such as *little* or *long*, inherently possess a degree of vagueness (Kamp and Partee, 1995; Kennedy, 2007): while in the extreme cases, judgments are strong (e.g., *a six foot tall woman* can clearly be called "a tall woman" whereas *a five foot tall woman* cannot), there are borderline cases for which it is difficult to say whether the adjectival predication can truthfully be ascribed to them. A logistic regression model can capture these facts. To build this model, we gather distributional information from the Web.

For instance, in the case of (3), we can retrieve from the Web positive and negative examples of age in relation to the adjective and the modified entity "little kids". The question contains the adjective and the modified entity. The reply contains the unit of measure (here "year-old") and the numerical answer. Specifically we query the Web using Yahoo! BOSS (Academic) for *"little kids" year-old* (positive instances) as well as for *"not little kids" year-old* (negative instances). Yahoo! BOSS is an open search services platform that provides a query API for Yahoo! Web search. We then extract ages from the positive and negative snippets obtained, and we fit a logistic regression to these data. To remove noise, we discard low counts (positive and negative instances for a given unit < 5). Also, for some adjectives, such as *little* or *young*, there is an inherent ambiguity between absolute and relative uses. Ideally, a word sense disambiguation system would be used to filter these cases. For now, we extract the largest contiguous range for which the data counts are over the noise threshold.[3] When not enough data is retrieved for the negative examples, we expand the query by moving the negation outside the search phrase. We also replace the negation and the adjective by the antonyms given in WordNet (using the first sense).

The logistic regression thus has only one factor — the unit of measure (age in the case of *little kids*). For a given answer, the model assigns a probability indicating the extent to which the adjectival property applies to that answer. If the factor is a significant predictor, we can use the probabilities from the model to decide whether the answer qualifies as a positive or negative instance of the adjective in the question, and thus interpret the indirect response as a 'yes' or a 'no'. The probabilistic nature of this technique adheres perfectly to the fact that indirect answers are intimately tied up with uncertainty.

5 Evaluation and results

Our primary goal is to evaluate how well we can learn the relevant scalar and entailment relationships from the Web. In the evaluation, we thus applied our techniques to a manually coded corpus version. For the adjectival scales, we annotated each example for its main predication (modifier, or adverb–modifier bigram), including whether that predication was negated. For the numerical cases, we manually constructed the initial queries: we identified the adjective and the modified entity in the question, and the unit of measure in the answer. However, we believe that identifying the requisite predications and recognizing the presence of negation or embedding could be done automatically using dependency graphs.[4]

[3] Otherwise, our model is ruined by references to "young 80-year olds", using the relative sense of *young*, which are moderately frequent on the Web.

[4] As a test, we transformed our corpus into the Stanford dependency representation (de Marneffe et al., 2006), using the Stanford parser (Klein and Manning, 2003) and were able to automatically retrieve all negated modifier predications, except one (*We had a view of it, not a particularly good one*),

	Modification in answer	Precision	Recall
I	Other adjective	60	60
	Adverb - same adjective	95	95
	Negation - same adjective	100	100
	Omitted adjective	100	100
II	Numerical	89	40
Total		75	71

Table 4: Summary of precision and recall (%) by type.

	Response	Precision	Recall	F1
I	Yes	87	76	81
	No	57	71	63
II	Yes	100	36	53
	Uncertain	67	40	50

Table 5: Precision, recall, and F1 (%) per response category. In the case of the scalar modifiers experiment, there were just two examples whose dominant response from the Turkers was 'Uncertain', so we have left that category out of the results. In the case of the numerical experiment, there were not any 'No' answers.

To evaluate the techniques, we pool the Mechanical Turk 'definite yes' and 'probable yes' categories into a single category 'Yes', and we do the same for 'definite no' and 'probable no'. Together with 'uncertain', this makes for three-response categories. We count an inference as successful if it matches the dominant Turker response category. To use the three-response scheme in the numerical experiment, we simply categorize the probabilities as follows: 0–0.33 = 'No', 0.33–0.66 = 'Uncertain', 0.66–1.00 = 'Yes'.

Table 4 gives a breakdown of our system's performance on the various category subtypes. The overall accuracy level is 71% (159 out of 224 inferences correct). Table 5 summarizes the results per response category, for the examples in which both the question and answer contain a gradable modifier (category I), and for the numerical cases (category II).

6 Analysis and discussion

Performance is extremely good on the "Adverb – same adjective" and "Negation – same adjective" cases because the 'Yes' answer is fairly direct for them (though adverbs like *basically* introduce an interesting level of uncertainty). The results are somewhat mixed for the "Other adjective" category.

Inferring the relation between scalar adjectives has some connection with work in sentiment detection. Even though most of the research in that domain focuses on the orientation of one term using seed sets, techniques which provide the orientation strength could be used to infer a scalar relation between adjectives. For instance, Blair-Goldensohn et al. (2008) use WordNet to develop sentiment lexicons in which each word has a positive or negative value associated with it, representing its strength. The algorithm begins with seed sets of positive, negative, and neutral terms, and then uses the synonym and antonym structure of WordNet to expand those initial sets and refine the relative strength values. Using our own seed sets, we built a lexicon using Blair-Goldensohn et al. (2008)'s method and applied it as in figure 3 (changing the ER values to sentiment scores). Both approaches achieve similar results: for the "Other adjective" category, the WordNet-based approach yields 56% accuracy, which is not significantly different from our performance (60%); for the other types in category I, there is no difference in results between the two methods. Table 6 summarizes the results per response category for the WordNet-based approach (which can thus be compared to the category I results in table 5). However in contrast to the WordNet-based approach, we require no hand-built resources: the synonym and antonym structures, as well as the strength values, are learned from Web data alone. In addition, the WordNet-based approach must be supplemented with a separate method for the numerical cases.

	Response	Precision	Recall	F1
WordNet-based	Yes	82	83	82.5
(items I)	No	60	56	58

Table 6: Precision, recall, and F1 (%) per response category for the WordNet-based approach.

In the "Other adjective" category, 31 items involve oppositional terms: canonical antonyms (e.g., *right*/*wrong*, *good*/*bad*) as well as terms that are "statistically oppositional" (e.g., *ready*/*premature*, *true*/*preposterous*, *confident*/*nervous*). "Statistically oppositional" terms are not oppositional by definition, but as a matter of contingent fact. Our technique accurately deals with most

because of a parse error which led to wrong dependencies.

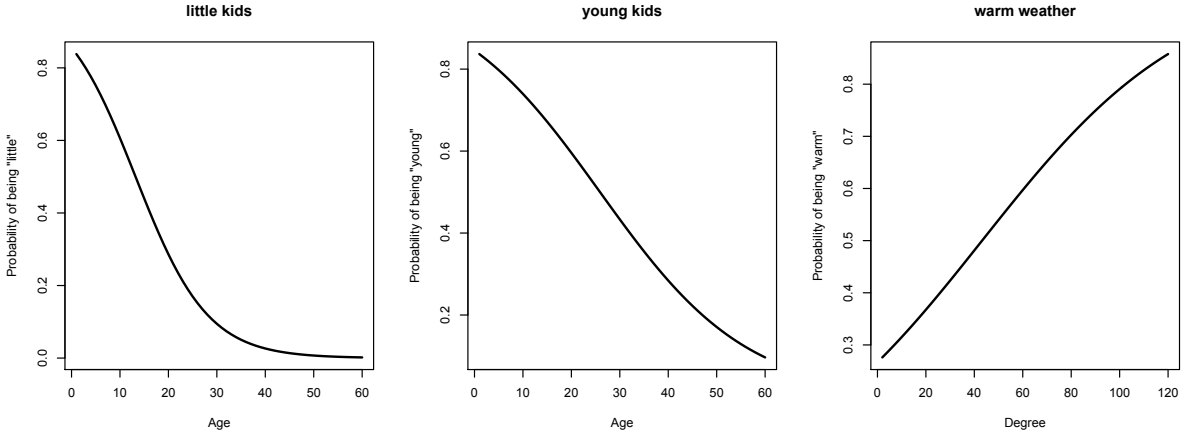


Figure 4: Probabilities of being appropriately described as “little”, “young” or “warm”, fitted on data retrieved when querying the Web for “little kids”, “young kids” and “warm weather”.

of the canonical antonyms, and also finds some contingent oppositions (*qualified/young*, *wise/ neurotic*) that are lacking in antonymy resources or automatically generated antonymy lists (Mohammad et al., 2008). Out of these 31 items, our technique correctly marks 18, whereas Mohammad et al.’s list of antonyms only contains 5 and Blair-Goldensohn et al. (2008)’s technique finds 11. Our technique is solely based on unigrams, and could be improved by adding context: making use of dependency information, as well as moving beyond unigrams.

In the numerical cases, precision is high but recall is low. For roughly half of the items, not enough negative instances can be gathered from the Web and the model lacks predictive power (as for items (4) or (12)).

(12) A: Do you happen to be working for a large firm?

B: It’s about three hundred and fifty people.

Looking at the negative hits for item (12), one sees that few give an indication about the number of people in the firm, but rather qualifications about colleagues or employees (*great people*, *people’s productivity*), or the hits are less relevant: “Most of the *people* I talked to were actually pretty optimistic. They were rosy on the job market and many had jobs, although most were *not large firm* jobs”. The lack of data comes from the fact that the queries are very specific, since the adjective depends on the product (e.g., “expensive exercise bike”, “deep pond”). However when we do get a predictive model, the probabilities corre-

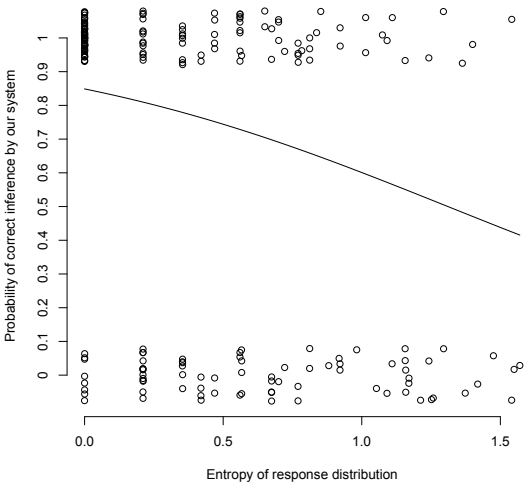


Figure 5: Correlation between agreement among Turkers and whether the system gets the correct answer. For each dialogue, we plot a circle at Turker response entropy and either 1 = correct inference or 0 = incorrect inference, except the points are jittered a little vertically to show where the mass of data lies. As the entropy rises (i.e., as agreement levels fall), the system’s inferences become less accurate. The fitted logistic regression model (black line) has a statistically significant coefficient for response entropy ($p < 0.001$).

late almost perfectly with the Turkers' responses. This happens for 8 items: "expensive to call (50 cents a minute)", "little kids (7 and 10 year-old)", "long growing season (3 months)", "lot of land (80 acres)", "warm weather (80 degrees)", "young kids (5 and 2 year-old)", "young person (31 year-old)" and "large house (2450 square feet)". In the latter case only, the system output (uncertain) doesn't correlate with the Turkers' judgment (where the dominant answer is 'probable yes' with 15 responses, and 11 answers are 'uncertain').

The logistic curves in figure 4 capture nicely the intuitions that people have about the relation between age and "little kids" or "young kids", as well as between Fahrenheit degrees and "warm weather". For "little kids", the probabilities of being little or not are clear-cut for ages below 7 and above 15, but there is a region of vagueness in between. In the case of "young kids", the probabilities drop less quickly with age increasing (an 18 year-old can indeed still be qualified as a "young kid"). In sum, when the data is available, this method delivers models which fit humans' intuitions about the relation between numerical measure and adjective, and can handle pragmatic inference.

If we restrict attention to the 66 examples on which the Turkers completely agreed about which of these three categories was intended (again pooling 'probable' and 'definite'), then the percentage of correct inferences rises to 89% (59 correct inferences). Figure 5 plots the relationship between the response entropy and the accuracy of our decision procedure, along with a fitted logistic regression model using response entropy to predict whether our system's inference was correct. The handful of empirical points in the lower left of the figure show cases of high agreement between Turkers but incorrect inference from the system. The few points in the upper right indicate low agreement between Turkers and correct inference from the system. Three of the high-agreement/incorrect-inference cases involve the adjectives *right–correct*. For low-agreement/correct-inference, the disparity could trace to context dependency: the ordering is clear in the context of product reviews, but unclear in a television interview. The analysis suggests that overall agreement is positively correlated with our system's chances of making a correct inference: our system's accuracy drops as human agreement levels drop.

7 Conclusion

We set out to find techniques for grounding basic meanings from text and enriching those meanings based on information from the immediate linguistic context. We focus on gradable modifiers, seeking to learn scalar relationships between their meanings and to obtain an empirically grounded, probabilistic understanding of the clear and fuzzy cases that they often give rise to (Kamp and Partee, 1995). We show that it is possible to learn the requisite scales between modifiers using review corpora, and to use that knowledge to drive inference in indirect responses. When the relation in question is not too specific, we show that it is also possible to learn the strength of the relation between an adjective and a numerical measure.

Acknowledgments

This paper is based on work funded in part by ONR award N00014-10-1-0109 and ARO MURI award 548106, as well as by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL), ARO or ONR.

References

James F. Allen and C. Raymond Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press, Cambridge.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A. Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era (NLPIX)*.

Judith A. Chevalier and Dina Mayzlin. 2006. The effect of word of mouth on sales: Online book reviews. *Journal of Marketing Research*, 43(3):345–354.

Herbert H. Clark. 1979. Responding to indirect speech acts. *Cognitive Psychology*, 11:430–477.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed

dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*.

Marie-Catherine de Marneffe, Scott Grimm, and Christopher Potts. 2009. Not a simple 'yes' or 'no': Uncertainty in indirect answers. In *Proceedings of the 10th Annual SIGDIAL Meeting on Discourse and Dialogue*.

Gilles Fauconnier. 1975. Pragmatic scales and logical structure. *Linguistic Inquiry*, 6(3):353–375.

Nancy Green and Sandra Carberry. 1994. A hybrid reasoning model for indirect answers. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 58–65.

Nancy Green and Sandra Carberry. 1999. Interpreting and generating indirect answers. *Computational Linguistics*, 25(3):389–435.

Beth Ann Hockey, Deborah Rossen-Knill, Beverly Spejewski, Matthew Stone, and Stephen Isard. 1997. Can you predict answers to Y/N questions? Yes, No and Stuff. In *Proceedings of Eurospeech 1997*, pages 2267–2270.

Laurence R Horn. 1972. *On the Semantic Properties of Logical Operators in English*. Ph.D. thesis, UCLA, Los Angeles.

Nan Hu, Paul A. Pavlou, and Jennifer Zhang. 2006. Can online reviews reveal a product's true quality?: Empirical findings and analytical modeling of online word-of-mouth communication. In *Proceedings of Electronic Commerce (EC)*, pages 324–330.

Daniel Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical Report 97-02, University of Colorado, Boulder Institute of Cognitive Science.

Hans Kamp and Barbara H. Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129–191.

Christopher Kennedy and Louise McNally. 2005. Scale structure and the semantic typology of gradable predicates. *Language*, 81(2):345–381.

Christopher Kennedy. 2007. Vagueness and grammar: The semantics of relative and absolute gradable adjectives. *Linguistics and Philosophy*, 30(1):1–45.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association of Computational Linguistics*.

Stephen C. Levinson. 2000. *Presumptive Meanings: The Theory of Generalized Conversational Implicature*. MIT Press, Cambridge, MA.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-2008)*.

Robert Munro, Steven Bethard, Victor Kuperman, Vicky Tzuyin Lai, Robin Melnick, Christopher Potts, Tyler Schnoebelen, and Harry Tily. 2010. Crowdsourcing and language studies: The new generation of linguistic data. In *NAACL 2010 Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1):1–135.

C. Raymond Perrault and James F. Allen. 1980. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3-4):167–182.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD-2008*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-2008)*.

Henk Zeevat. 1994. Questions and exhaustivity in update semantics. In Harry Bunt, Reinhard Muskens, and Gerrit Rentier, editors, *Proceedings of the International Workshop on Computational Semantics*, pages 211–221.

Importance-Driven Turn-Bidding for Spoken Dialogue Systems

Ethan O. Selfridge and Peter A. Heeman
Center for Spoken Language Understanding
Oregon Health & Science University
20000 NW Walker Rd., Beaverton, OR, 97006

selfridg@ohsu.edu, heemanp@ohsu.edu

Abstract

Current turn-taking approaches for spoken dialogue systems rely on the speaker releasing the turn before the other can take it. This reliance results in restricted interactions that can lead to inefficient dialogues. In this paper we present a model we refer to as *Importance-Driven Turn-Bidding* that treats turn-taking as a negotiative process. Each conversant bids for the turn based on the importance of the intended utterance, and Reinforcement Learning is used to indirectly learn this parameter. We find that Importance-Driven Turn-Bidding performs better than two current turn-taking approaches in an artificial collaborative slot-filling domain. The negotiative nature of this model creates efficient dialogues, and supports the improvement of mixed-initiative interaction.

1 Introduction

As spoken dialogue systems are designed to perform ever more elaborate tasks, the need for mixed-initiative interaction necessarily grows. Mixed-initiative interaction, where agents (both artificial and human) may freely contribute to reach a solution efficiently, has long been a focus of dialogue systems research (Allen et al., 1999; Guinn, 1996). Simple slot-filling tasks might not require the flexible environment that mixed-initiative interaction brings but those of greater complexity, such as collaborative task completion or long-term planning, certainly do (Ferguson et al., 1996). However, translating this interaction into working systems has proved problematic (Walker et al., 1997), in part to issues surrounding turn-taking: the transition from one speaker to another.

Many computational turn-taking approaches seek to minimize silence and utterance overlap during transitions. This leads to the speaker controlling the turn transition. For example, systems using the Keep-Or-Release approach will not attempt to take the turn unless it is sure the user has released it. One problem with this approach is that the system might have important information to give but will be unable to get the turn. The speaker-centric nature of current approaches does not enable mixed-initiative interaction and results in inefficient dialogues. Primarily, these approaches have been motivated by smooth transitions reported in the human turn-taking studies of Sacks et al. (1974) among others.

Sacks et al. also acknowledge the negotiative nature of turn-taking, stating that the "the turn as unit is interactively determined"(p. 727). Other studies have supported this, suggesting that humans negotiate the turn assignment through the use of cues and that these cues are motivated by the importance of what the conversant wishes to contribute (Duncan and Niederehe, 1974; Yang and Heeman, 2010; Schegloff, 2000). Given this, any dialogue system hoping to interact with humans efficiently and naturally should have a negotiative and importance-driven quality to its turn-taking protocol. We believe that, by focusing on the rationale of human turn-taking behavior, a more effective turn-taking system may be achieved. We propose the Importance-Driven Turn-Bidding (IDTB) model where conversants bid for the turn based on the importance of their utterance. We use Reinforcement Learning to map a given situation to the optimal utterance and bidding behavior. By allowing conversants to bid for the turn, the IDTB model enables negotiative turn-taking and supports true mixed-initiative interaction, and with it, greater dialogue efficiency.

We compare the IDTB model to current turn-taking approaches. Using an artificial collaborative dialogue task, we show that the IDTB model enables the system and user to complete

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 177–185,
Uppsala, Sweden, 11-16 July 2010.

the task more efficiently than the other approaches. Though artificial dialogues are not ideal, they allow us to test the validity of the IDTB model before embarking on costly and time-consuming human studies. Since our primary evaluation criteria is model comparison, consistent user simulations provide a constant needed for such measures and increase the external validity of our results.

2 Current Turn-Taking Approaches

Current dialogue systems focus on the release-turn as the most important aspect of turn-taking, in which a listener will only take the turn after the speaker has released it. The simplest of these approaches only allows a single utterance per turn, after which the turn necessarily transitions to the next speaker. This Single-Utterance (SU) model has been extended to allow the speaker to keep the turn for multiple utterances: the Keep-Or-Release (KR) approach. Since the KR approach gives the speaker sole control of the turn, it is overwhelmingly speaker-centric, and so necessarily unnegotiative. This restriction is meant to encourage smooth turn-transitions, and is inspired by the order, smoothness, and predictability reported in human turn-taking studies (Duncan, 1972; Sacks et al., 1974).

Systems using the KR approach differ on how they detect the user's release-turn. Turn releases are commonly identified in two ways: either using a silence-threshold (Sutton et al., 1996), or the predictive nature of turn endings (Sacks et al., 1974) and the cues associated with them (e.g. Gravano and Hirschberg, 2009). Raux and Eskenazi (2009) used decision theory with lexical cues to predict appropriate places to take the turn. Similarly, Jonsdottir, Thorisson, and Nivel (2008) used Reinforcement Learning to reduce silences between turns and minimize overlap between utterances by learning the specific turn-taking patterns of individual speakers. Skantze and Schlangan (2009) used incremental processing of speech and prosodic turn-cues to reduce the reaction time of the system, finding that that users rated this approach as more human-like than a baseline system.

In our view, systems built using the KR turn-taking approach suffer from two deficits. First, the speaker-centricity leads to inefficient dialogues since the speaker may continue to hold the turn even when the listener has vital information to give. In addition, the lack of negotiation forces the turn to necessarily transition to the listener after the speaker releases it. The possibility that the dialogue may be better served if the listener *does not* get the turn is not addressed by current approaches.

Barge-in, which generally refers to allowing users to speak at any time (Ström and Seneff, 2000), has been the primary means to create a more flexible turn-taking environment. Yet, since barge-in recasts speaker-centric systems as user-centric, the system's contributions continue to be limited. System barge-in has also been investigated. Sato et al. (2002) used decision trees to determine whether the system should take the turn or not when the user pauses. An incremental method by DeVault, Sagae, and Traum (2009) found possible points that a system could interrupt without loss of user meaning, but failed to supply a reasonable model as to when to use such information. Despite these advances, barge-in capable systems lack a negotiative turn-taking method, and continue to be deficient for reasons similar to those described above.

3 Importance-Driven Turn-Bidding (IDTB)

We introduce the IDTB model to overcome the deficiencies of current approaches. The IDTB model has two foundational components: (1) The importance of speaking is the primary motivation behind turn-taking behavior, and (2) conversants use turn-cue strength to bid for the turn based on this importance. Importance may be broadly defined as how well the utterance leads to some predetermined conversational success, be it solely task completion or encompassing a myriad of social etiquette components.

Importance-Driven Turn-Bidding is motivated by empirical studies of human turn-conflict resolution. Yang and Heeman (2010) found an increase of turn conflicts during tighter time constraints, which suggests that turn-taking is influenced by the importance of task completion. Schlegoff (2000) proposed that persistent utterance overlap was indicative of conversants having a strong interest in holding the turn. Walker and Whittaker (1990) show that people will interrupt to remedy some understanding discrepancy, which is certainly important to the conversation's success. People communicate the importance of their utterance through turn-cues. Duncan and

Niederehe (1974) found that turn-cue strength was the best predictor of who won the turn, and this finding is consistent with the use of volume to win turns found by Yang and Heeman (2010).

The IDTB model uses turn-cue strength to bid for the turn based on the importance of the utterance. Stronger turn-cues should be used when the intended utterance is important to the overall success of the dialogue, and weaker ones when it is not. In the prototype described in Section 5, both the system and user agents bid for the turn after every utterance and the bids are conceptualized here as utterance onset: conversants should be quick to speak important utterances but slow with less important ones. This is relatively consistent with Yang and Heeman (2010). A mature version of our work will use cues in addition to utterance onset, such as those recently detailed in Gravano and Hirshberg (2009).[1]

A crucial element of our model is the judgment and quantization of utterance importance. We use Reinforcement Learning (RL) to determine importance by conceptualizing it as maximizing the reward over an entire dialogue. Whatever actions lead to a higher return may be thought of as more important than ones that do not.[2] By using RL to learn both the utterance and bid behavior, the system can find an optimal pairing between them, and choose the best combination for a given conversational situation.

4 Information State Update and Reinforcement Learning

We build our dialogue system using the Information State Update approach (Larsson and Traum, 2000) and use Reinforcement Learning for action selection (Sutton and Barto, 1998). The system architecture consists of an Information State (IS) that represents the agent's knowledge and is updated using a variety of rules. The IS also uses rules to propose possible actions. A condensed and compressed subset of the IS — the Reinforcement Learning State — is used to learn which proposed action to take (Heeman, 2007). It has been shown that using RL to learn dialogue polices is generally more effective than "hand crafted" dialogue policies since the learning algorithm may capture environmental dynamics that are unattended to by human designers (Levin et al., 2000).

Reinforcement Learning learns an optimal policy, a mapping between a state s and action a, where performing a in s leads to the lowest expected cost for the dialogue (we use minimum cost instead of maximum reward). An *ϵ-greedy* search is used to estimate Q-scores, the expected cost of some state–action pair, where the system chooses a random action with ϵ probability and the $\mathit{argmin}_a Q(s,a)$ action with 1-ϵ probability. For Q-learning, a popular RL algorithm and the one used here, ϵ is commonly set at 0.2 (Sutton and Barto, 1998). Q-learning updates $Q(s,a)$ based on the best action of the next state, given by the following equation, with the step size parameter $\alpha = 1/\sqrt{N(s,a)}$ where $N(s,a)$ is the number of times the s,a pair has been seen since the beginning of training.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[\mathit{cost}_{t+1} + \mathit{argmin}_a Q(s_{t+1},a) - Q(s_t,a_t)]$$

The state space should be formulated as a Markov Decision Process (MDP) for Q-learning to update Q-scores properly. An MDP relies on a first-order Markov assumption in that the transition and reward probability from some s_t, a_t pair is completely contained by that pair and is unaffected by the history $s_{t-1}a_{t-1}, s_{t-2}a_{t-2}, \ldots$. For this assumption to be met, care is required when deciding which features to include for learning. The RL State features we use are described in the following section.

5 Domain and Turn-Taking Models

In this section, we show how the IDTB approach can be implemented for a collaborative slot filling domain. We also describe the Single-Utterance and Keep-Or-Release domain implementations that we use for comparison.

5.1 Domain Task

We use a food ordering domain with two participants, the system and a user, and three slots: drink, burger, and side. The system's objective is to fill all three slots with the available fillers as quickly as possible. The user's role is to specify its desired filler for each slot, though that specific filler may not be available. The user simulation, while intended to be realistic, is not based on empirical data. Rather, it is designed to provide a rich turn-

[1] Our work (present and future) is distinct from some recent work on user pauses (Sato et al., 2002) since we treat turn-taking as an integral piece of dialogue success.

[2] We gain an inherent flexibility in using RL since the reward can be computed by a wide array of components. This is consistent with the broad definition of importance.

taking domain to evaluate the performance of different turn-taking designs. We consider this a collaborative slot-filling task since both conversants must supply information to determine the intersection of available and desired fillers.

Users have two fillers for each slot.[3] A user's top choice is either available, in which case we say that the user has adequate filler knowledge, or their second choice will be available, in which we say it has inadequate filler knowledge. This assures that at least one of the user's filler is available. Whether a user has adequate or inadequate filler knowledge is probabilistically determined based on user type, which will be described in Section 5.2.

Table 1: Agent speech acts

Agent	Actions
System	query slot, inform [yes/no], inform avail. slot fillers, inform filler not available, bye
User	inform slot filler, query filler availability

We model conversations at the speech act level, shown in Table 1, and so do not model the actual words that the user and system might say. Each agent has an Information State that proposes possible actions. The IS is made up of a number of variables that model the environment and is slightly different for the system and the user. Shared variables include *QUD*, a stack which manages the questions under discussion; *lastUtterance*, the previous utterance, and *slotList*, a list of the slot names. The major system specific IS variables that are not included in the RL State are *availSlotFillers*, the available fillers for each slot; and three *slotFiller* variables that hold the fillers given by the user. The major user specific IS variables are three *desiredSlotFiller* variables that hold an ordered list of fillers, and *unvisitedSlots*, a list of slots that the user believes are unfilled.

The system has a variety of speech actions: *inform [yes/no]*, to answer when the user has asked a filler availability question; *inform filler not available*, to inform the user when they have specified an unavailable filler; three *query slot* actions (one for each slot), a query which asks the user for a filler and is proposed if that specific slot is unfilled; three *inform available slot fillers* actions, which lists the available fillers for that slot and is proposed if that specific slot is unfilled or filled with an unavailable filler; and *bye*, which is always proposed.

The user has two actions. They can inform the system of a desired slot filler, *inform slot filler*, or query the availability of a slot's top filler, *query filler availability*. A user will always respond with the same slot as a system query, but may change slots entirely for all other situations. Additional details on user action selection are given in Section 5.2.

Specific information is used to produce an instantiated speech action, what we refer to as an utterance. For example, the speech action *inform slot filler* results in the utterance of "inform drink d1." A sample dialogue fragment using the Single-Utterance approach is shown in Table 2. Notice that in Line 3 the system informs the user that their first filler, *d1*, is unavailable. The user then asks asks about the availability of its second drink choice, *d2* (Line 4), and upon receiving an affirmative response (Line 5), informs the system of that filler preference (Line 6).

Table 2: Single-Utterance dialogue

Spkr	Speech Action	Utterance
1 S:	*q. slot*	q. drink
2 U:	*i. slot filler*	i. drink d1
3 S:	*i. filler not avail*	i. not have d1
4 U:	*q. filler avail*	q. drink have d2
5 S:	*i. slot*	i. yes
6 U:	*i. slot filler*	i. drink d2
7 S:	*i. avail slot fillers*	i. burger have b1

Implementation in RL: The system uses RL to learn which of the IS proposed actions to take. In this domain we use a cost function based on dialogue length and the number of slots filled with an available filler: $C = \text{Number of Utterances} + 25 \cdot \text{unavailablyFilledSlots}$. In the present implementation the system's *bye* utterance is costless. The system chooses the action that minimizes the expected cost of the entire dialogue from the current state.

The RL state for the speaker has seven variables:[4] *QUD-speaker*, the stack of speakers who have unresolved questions; *Incorrect-Slot-Fillers*,

[3] We use two fillers so as to minimize the length of training. This can be increased without substantial effort.

[4] We experimented with a variety of RL States and this one proved to be both small and effective.

a list of slot fillers (ordered chronologically on when the user informed them) that are unavailable and have not been resolved; *Last-Sys-Speech-Action*, the last speech action the system performed; *Given-Slot-Fillers*, a list of slots that the system has performed the *inform available slot filler* action on; and three booleans variables, *slot-RL*, that specify whether a slot has been filled correctly or not (e.g. Drink-RL).

5.2 User Types

We define three different types of users — *Experts*, *Novices*, and *Intermediates*. User types differ probabilistically on two dimensions: slot knowledge, and slot belief strength. We define experts to have a 90 percent chance of having adequate filler knowledge, intermediates a 50 percent chance, and novices a 10 percent chance. These probabilities are independent between slots. Slot belief strength represents the user's confidence that it has adequate domain knowledge for the slot (i.e. the top choice for that slot is available). It is either a strong, warranted, or weak belief (Chu-Carroll and Carberry, 1995). The intuition is that experts should know when their top choice is available, and novices should know that they do not know the domain well.

Initial slot belief strength is dependent on user type and whether their filler knowledge is adequate (their initial top choice is available). Experts with adequate filler knowledge have a 70, 20, and 10 percent chance of having Strong, Warranted, and Weak beliefs respectfully. Similarly, intermediates with adequate knowledge have a 50, 25, and 25 percent chance of the respective belief strengths. When these user types have inadequate filler knowledge the probabilities are reversed to determine belief strength (e.g. Experts with inadequate domain knowledge for a slot have a 70% chance of having a weak belief). Novice users always have a 10, 10, and 80 percent chance of the respective belief strengths.

The user choses whether to use the *query* or *inform* speech action based on the slot's belief strength. A strong belief will always result in an *inform*, a warranted belief resulting in an *inform* with $p = 0.5$, and weak belief will result in an *inform* with $p = 0.25$. If the user is informed of the correct fillers by the system's *inform*, that slot's belief strength is set to strong. If the user is informed that a filler is not available, than that filler is removed from the desired filler list and the belief remains the same.[5]

5.3 Turn-Taking Models

We now discuss how turn-taking works for the IDTB model and the two competing models that we use to evaluate our approach. The system chooses its turn action based on the RL state and we add a boolean variable *turn-action* to the RL State to indicate when the system is performing a turn action or a speech action. The user uses belief to choose its turn action.

Turn-Bidding: Agents bid for the turn at the end of each utterance to determine who will speak next. Each bid is represented as a value between 0 and 1, and the agent with the lower value (stronger bid) wins the turn. This is consistent with the use of utterance onset. There are 5 types of bids, *highest, high, middle, low,* and *lowest*, which are spread over a portion of the range as shown in Figure 1. The system uses RL to choose a bid and a random number (uniform distribution) is generated from that bid's range. The users' bids are determined by their belief strength, which specifies the mean of a Gaussian distribution, as shown in Figure 1 (e.g Strong belief implies a μ = 0.35). Computing bids in this fashion leads to, on average, users with strong beliefs bidding *highest*, warranted beliefs bidding in the *middle*, and weak beliefs bidding *lowest*. The use of the probability distributions allows us to randomly decide ties between system and user bids.

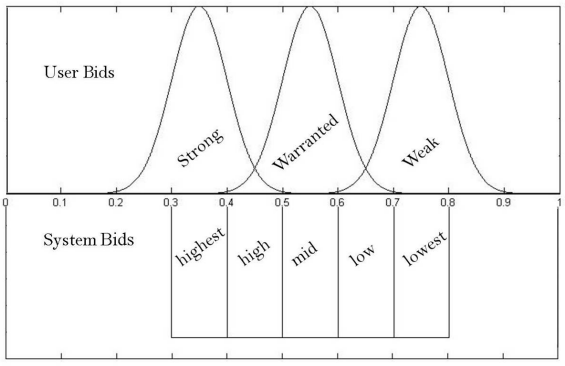


Figure 1: Bid Value Probability Distribution

Single-Utterance: The Single-Utterance (SU) approach, as described in Section 2, has a rigid

[5] In this simple domain the next filler is guaranteed to be available if the first is not. We do not model this with belief strength since it is probably not representative of reality.

turn-taking mechanism. After a speaker makes a single utterance the turn transitions to the listener. Since the turn transitions after every utterance the system must only choose appropriate utterances, not turn-taking behavior. Similarly, user agents do not have any turn-taking behavior and slot beliefs are only used to choose between a *query* and an *inform*.

Keep-Or-Release Model: The Keep-Or-Release (KR) model, as described in Section 2, allows the speaker to either keep the turn to make multiple utterances or release it. Taking the same approach as English and Heeman (2005), the system learns to keep or release the turn after each utterance that it makes. We also use RL to determine which conversant should begin the dialogue. While the use of RL imparts some importance onto the turn-taking behavior, it is not influencing whether the system gets the turn when it did not already have it. This is an crucial distinction between KR and IDTB. IDTB allows the conversants to negotiate the turn using turn-bids motivated by importance, whereas in KR only the speaker determines when the turn can transition.

Users in the KR environment choose whether to keep or release the turn similarly to bid decisions.[6] After a user performs an utterance, it chooses the slot that would be in the next utterance. A number, k, is generated from a Gaussian distribution using belief strength in the same manner as the IDTB users' bids are chosen. If $k \leq 0.55$ then the user keeps the turn, otherwise it releases it.

5.4 Preliminary Turn-Bidding System

We described a preliminary turn-bidding system in earlier work presented at a workshop (Selfridge and Heeman, 2009). A major limitation was an overly simplified user model. We used two user types, expert and novice, who had fixed bids. Experts always bid *high* and had complete domain knowledge, and the novices always bid *low* and had incomplete domain knowledge. The system, using all five bid types, was always able to out bid and under bid the simulated users. Among other things, this situation gives the system complete control of the turn, which is at odds with the negotiative nature of IDTB. The present contribution is a more realistic and mature implementation.

[6]We experimented with a few different KR decision strategies, and chose the one that performed the best.

6 Evaluation and Discussion

We now evaluate the IDTB approach by comparing it against the two competing models: Single-Utterance and Keep-Or-Release. The three turn-taking approaches are trained and tested in four user conditions: novice, intermediate, expert, and combined. In the combined condition, one of the three user types is randomly selected for each dialogue. We train ten policies for each condition and turn-taking approach. Policies are trained using Q-learning, and $\epsilon-greedy$ search for 10000 epochs (1 epoch = 100 dialogues, after which the Q-scores are updated) with $\epsilon = 0.2$. Each policy is then ran over 10000 test dialogues with no exploration ($\epsilon = 0$), and the mean dialogue cost for that policy is determined. The 10 separate policy values are then averaged to create the mean policy cost. The mean policy cost between the turn-taking approaches and user conditions are shown in Table 3. Lower numbers are indicative of shorter dialogues, since the system learns to successfully complete the task in all cases.

Table 3: Mean Policy Cost for Model and User condition[7]

Model	Novice	Int.	Expert	Combined
SU	7.61	7.09	6.43	7.05
KR	**6.00**	6.35	4.46	6.01
IDTB	6.09	**5.77**	**4.35**	**5.52**

Single User Conditions: Single user conditions show how well each turn-taking approach can optimize its behavior for specific user populations and handle slight differences found in those populations. Table 3 shows that the mean policy cost of the SU model is higher than the other two models which indicates longer dialogues on average. Since the SU system must respond to every user utterance and cannot learn a turn-taking strategy to utilize user knowledge, the dialogues are necessarily longer. For example, in the expert condition the best possible dialogue for a SU interaction will have a cost of five (three user utterances for each slot, two system utterances in response). This cost is in contrast to the best expert dialogue cost of three (three user utterances) for KR and IDTB interactions.

The IDTB turn-taking approach outperforms the KR design in all single user conditions ex-

[7]SD between policies ≤ 0.04

cept for novice (6.09 vs. 6.00). In this condition, the KR system takes the turn first, informs the available fillers for each slot, and then releases the turn. The user can then inform its filler easily. The IDTB system attempts a similar dialogue strategy by using *highest* bids but sometimes loses the turn when users also bid *highest*. If the user uses the turn to query or inform an unavailable filler the dialogue grows longer. However, this is quite rare as shown by small difference in performance between the two models. In all other single user conditions, the IDTB approach has shorter dialogues than the KR approach (5.77 and 4.35 vs. 6.35 and 4.46). A detailed explanation of IDTB's performance will be given in Section 6.1.

Combined User Condition: We next measure performance on the combined condition that mixes all three user types. This condition is more realistic than the other three, as it better mimics how a system will be used in actual practice. The IDTB approach (mean policy cost = 5.52) outperforms the KR (mean policy cost = 6.01) and SU (mean policy cost = 7.05) approaches. We also observe that KR outperforms SU. These results suggest that the more a turn-taking design can be flexible and negotiative, the more efficient the dialogues can be.

Exploiting User bidding differences: It follows that IDTB's performance stems from its negotiative turn transitions. These transitions are distinctly different than KR transitions in that there is information inherent in the users bids. A user that has a stronger belief strength is more likely to be have a higher bid and inform an available filler. Policy analysis shows that the IDTB system takes advantage of this information by using moderate bids —neither *highest* nor *lowest* bids— to filter users based on their turn behavior. The distribution of bids used over the ten learned policies is shown in Table 4. The *initial* position refers to the first bid of the dialogue; *final* position, the last bid of the dialogue; and *medial* position, all other bids. Notice that the system uses either the *low* or *mid* bids as its *initial* policy and that 67.2% of dialogue *medial* bids are moderate. These distributions show that the system has learned to use the entire bid range to filter the users, and is not seeking to win or lose the turn outright. This behavior is impossible in the KR approach.

Table 4: Bid percentages over ten policies in the Combined User condition for IDTB

Position	H-est	High	Mid	Low	L-est
Initial	0.0	0.0	70.0	30.0	0.0
Medial	20.5	19.4	24.5	23.3	12.3
Final	49.5	41.0	9.5	0.0	0.0

6.1 IDTB Performance:

In our domain, performance is measured by dialogue length and solution quality. However, since solution quality never affects the dialogue cost for a trained system, dialogue length is the only component influencing the mean policy cost.

The primary cause of longer dialogues are unavailable filler inform and query (UFI–Q) utterances by the user, which are easily identified. These utterances lengthen the dialogue since the system must inform the user of the available fillers (the user would otherwise not know that the filler was unavailable) and then the user must then inform the system of its second choice. The mean number of UFI–Q utterance for each dialogue over the ten learned policies are shown for all user conditions in Table 5. Notice that these numbers are inversely related to performance: the more UFI–Q utterances, the worse the performance. For example, in the combined condition the IDTB users perform 0.38 UFI–Q utterances per dialogue (u/d) compared to the 0.94 UFI–Q u/d for KR users. While a KR user will release the turn if its planned utterance has a weak belief, it may select that weak utterance when first getting the turn (either after a system utterance or at the start of the dialogue). This may lead to a UFI–Q utterance. The IDTB system, however, will outbid the same user, resulting in a shorter dialogue. This situation is shown in Tables 6 and 7. The dialogue is the same until utterance 3, where the IDTB system wins the turn with a *mid* bid over the user's *low* bid. In the KR environment however, the user gets the turn and performs an unavailable filler inform, which the system must react to. This is an instance of the second deficiency of the KR approach, where

Table 5: Mean number of UFI–Q utterances over policies

Model	Novice	Int.	Expert	Combined
KR	**0.0**	1.15	0.53	0.94
IDTB	0.1	**0.33**	**0.39**	**0.38**

Table 6: Sample IDTB dialogue in Combined User condition; Cost=6

Sys	Usr	Spkr	Utt
1 low	mid	U:	inform burger b1
2 h-est	low	S:	inform burger have b3
3 mid	low	S:	inform side have s1
4 mid	h-est	U:	inform burger b3
5 mid	high	U:	inform drink d1
6 l-est	h-est	U:	inform side s1
7 high	mid	S:	bye

Table 7: Sample KR dialogue in Combined User condition; Cost=7

Agent	Utt	Turn-Action
1 U:	inform burger b1	Release
2 S:	inform burger have b3	Release
3 U:	inform side s1	Keep
4 U:	inform drink d1	Keep
5 U:	inform burger b3	Release
6 S:	inform side have s2	Release
7 U:	inform side s2	Release
8 S:	bye	

the speaking system should not have released the turn. The user has the same belief in both scenarios, but the negotiative nature of IDTB enables a shorter dialogues. In short, the IDTB system can win the turn when it should have it, but the KR system cannot.

A lesser cause of longer dialogues is an instance of the first deficiency of the KR systems; the listening user cannot get the turn when it should have it. Usually, this situation presents itself when the user releases the turn, having randomly chosen the weaker of the two unfilled slots. The system then has the turn for more than one utterance, informing the available fillers for two slots. However, the user already had a strong belief and available top filler for one of those slots, and the system has increased the dialogue length unnecessarily. In the combined condition, the KR system produces 0.06 unnecessary informs per dialogue, whereas the IDTB system produces 0.045 per dialogue. The novice and intermediate conditions mirror this (IDTB: 0.009, 0.076 ; KR: 0.019, 0.096 respectfully), but the expert condition does not (IDTB: 0.011, KR: 0.0014). In this case, the IDTB system wins the turn initially using a *low* bid and informs one of the strong slots, whereas the expert user initiates the dialogue for the KR environment and unnecessary informs are rarer. In general, however, the KR approach has more unnecessary informs since the KR system can only infer that one of the user's beliefs was probably weak, otherwise the user would not have released the turn. The IDTB system handles this situation by using a *high* bid, allowing the user to outbid the system as its contribution is more important. In other words, the IDTB user can win the turn when it should have it, but the KR user cannot.

7 Conclusion

This paper presented the Importance-Driven Turn-Bidding model of turn-taking. The IDTB model is motivated by turn-conflict studies showing that the interest in holding the turn influences conversant turn-cues. A computational prototype using Reinforcement Learning to choose appropriate turn-bids performs better than the standard KR and SU approaches in an artificial collaborative dialogue domain. In short, the Importance-Driven Turn-Bidding model provides a negotiative turn-taking framework that supports mixed-initiative interactions.

In the previous section, we showed that the KR approach is deficient for two reasons: the speaking system might not keep the turn when it should have, and might release the turn when it should not have. This is driven by KR's speaker-centric nature; the speaker has no way of judging the potential contribution of the listener. The IDTB approach however, due to its negotiative quality, does not have this problem.

Our performance differences arise from situations when the system is the speaker and the user is the listener. The IDTB model also excels in the opposite situation, when the system is the listener and the user is the speaker, though our domain is not sophisticated enough for this situation to occur. In the future we hope to develop a domain with more realistic speech acts and a more difficult dialogue task that will, among other things, highlight this situation. We also plan on implementing a fully functional IDTB system, using an incremental processing architecture that not only detects, but generates, a wide array of turn-cues.

Acknowledgments

We gratefully acknowledge funding from the National Science Foundation under grant IIS-0713698.

References

J.E Allen, C.I. Guinn, and Horvitz E. 1999. Mixed-initiative interaction. *IEEE Intelligent Systems*, 14(5):14–23.

Jennifer Chu-Carroll and Sandra Carberry. 1995. Response generation in collaborative negotiation. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 136–143, Morristown, NJ, USA. Association for Computational Linguistics.

David DeVault, Kenji Sagae, and David Traum. 2009. Can i finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference*, pages 11–20, London, UK, September. Association for Computational Linguistics.

S.J. Duncan and G. Niederehe. 1974. On signalling that it's your turn to speak. *Journal of Experimental Social Psychology*, 10:234–247.

S.J. Duncan. 1972. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23:283–292.

M. English and Peter A. Heeman. 2005. Learning mixed initiative dialog strategies by using reinforcement learning on both conversants. In *Proceedings of HLT/EMNLP*, pages 1011–1018.

G. Ferguson, J. Allen, and B. Miller. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77.

A. Gravano and J. Hirschberg. 2009. Turn-yielding cues in task-oriented dialogue. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 253–261. Association for Computational Linguistics.

C.I. Guinn. 1996. Mechanisms for mixed-initiative human-computer collaborative discourse. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 278–285. Association for Computational Linguistics.

P.A. Heeman. 2007. Combining reinforcement learning with information-state update rules. In *Proceedings of the Annual Conference of the North American Association for Computational Linguistics*, pages 268–275, Rochester, NY.

Gudny Ragna Jonsdottir, Kristinn R. Thorisson, and Eric Nivel. 2008. Learning smooth, human-like turntaking in realtime dialogue. In *IVA '08: Proceedings of the 8th international conference on Intelligent Virtual Agents*, pages 162–175, Berlin, Heidelberg. Springer-Verlag.

S. Larsson and D. Traum. 2000. Information state and dialogue managment in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6:323–340.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11 – 23.

A. Raux and M. Eskenazi. 2009. A finite-state turn-taking model for spoken dialog systems. In *Proceedings of HLT/NAACL*, pages 629–637. Association for Computational Linguistics.

H. Sacks, E.A. Schegloff, and G. Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

R. Sato, R. Higashinaka, M. Tamoto, M. Nakano, and K. Aikawa. 2002. Learning decision trees to determine turn-taking by spoken dialogue systems. In *ICSLP*, pages 861–864, Denver, CO.

E.A. Schegloff. 2000). Overlapping talk and the organization of turn-taking for conversation. *Language in Society*, 29:1 – 63.

E. O. Selfridge and Peter A. Heeman. 2009. A bidding approach to turn-taking. In *1st International Workshop on Spoken Dialogue Systems*.

G. Skantze and D. Schlangen. 2009. Incremental dialogue processing in a micro-domain. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 745–753. Association for Computational Linguistics.

N. Ström and S. Seneff. 2000. Intelligent barge-in in conversational systems. In *Sixth International Conference on Spoken Language Processing*. Citeseer.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press.

S. Sutton, D. Novick, R. Cole, P. Vermeulen, J. de Villiers, J. Schalkwyk, and M. Fanty. 1996. Building 10,000 spoken-dialogue systems. In *ICSLP*, Philadelphia, Oct.

M. Walker and S. Whittaker. 1990. Mixed initiative in dialoge: an investigation into discourse segmentation. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 70–76.

M. Walker, D. Hindle, J. Fromer, G.D. Fabbrizio, and C. Mestel. 1997. Evaluating competing agent strategies for a voice email agent. In *Fifth European Conference on Speech Communication and Technology*.

Fan Yang and Peter A. Heeman. 2010. Initiative conflicts in task-oriented dialogue". *Computer Speech Language*, 24(2):175 – 189.

Entity-based local coherence modelling using topological fields

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

Abstract

One goal of natural language generation is to produce coherent text that presents information in a logical order. In this paper, we show that topological fields, which model high-level clausal structure, are an important component of local coherence in German. First, we show in a sentence ordering experiment that topological field information improves the entity grid model of Barzilay and Lapata (2008) more than grammatical role and simple clausal order information do, particularly when manual annotations of this information are not available. Then, we incorporate the model enhanced with topological fields into a natural language generation system that generates constituent orders for German text, and show that the added coherence component improves performance slightly, though not statistically significantly.

1 Introduction

One type of coherence modelling that has captured recent research interest is local coherence modelling, which measures the coherence of a document by examining the similarity between neighbouring text spans. The entity-based approach, in particular, considers the occurrences of noun phrase entities in a document (Barzilay and Lapata, 2008). Local coherence modelling has been shown to be useful for tasks like natural language generation and summarization, (Barzilay and Lee, 2004) and genre classification (Barzilay and Lapata, 2008).

Previous work on English, a language with relatively fixed word order, has identified factors that contribute to local coherence, such as the grammatical roles associated with the entities. There is good reason to believe that the importance of these factors vary across languages. For instance, freer-word-order languages exhibit word order patterns which are dependent on discourse factors relating to information structure, in addition to the grammatical roles of nominal arguments of the main verb. We thus expect word order information to be particularly important in these languages in discourse analysis, which includes coherence modelling.

For example, Strube and Hahn (1999) introduce *Functional Centering*, a variant of Centering Theory which utilizes information status distinctions between hearer-old and hearer-new entities. They apply their model to pronominal anaphora resolution, identifying potential antecedents of subsequent anaphora by considering syntactic and word order information, classifying constituents by their familiarity to the reader. They find that their approach correctly resolves more pronominal anaphora than a grammatical role-based approach which ignores word order, and the difference between the two approaches is larger in German corpora than in English ones. Unfortunately, their criteria for ranking potential antecedents require complex syntactic information in order to classify whether proper names are known to the hearer, which makes their algorithm hard to automate. Indeed, all evaluation is done manually.

We instead use topological fields, a model of clausal structure which is indicative of information structure in German, but shallow enough to be automatically parsed at high accuracy. We test the hypothesis that they would provide a good complement or alternative to grammatical roles in local coherence modelling. We show that they are superior to grammatical roles in a sentence ordering experiment, and in fact outperforms simple word-order information as well. We further show that these differences are particularly large when manual syntactic and grammatical role an-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 186–195,
Uppsala, Sweden, 11-16 July 2010.

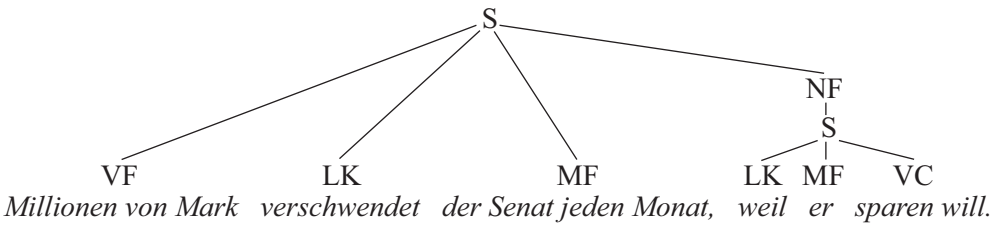


Figure 1: The clausal and topological field structure of a German sentence. Notice that the subordinate clause receives its own topology.

notations are not available.

We then embed these topological field annotations into a natural language generation system to show the utility of local coherence information in an applied setting. We add contextual features using topological field transitions to the model of Filippova and Strube (2007b) and achieve a slight improvement over their model in a constituent ordering task, though not statistically significantly. We conclude by discussing possible reasons for the utility of topological fields in local coherence modelling.

2 Background and Related Work

2.1 German Topological Field Parsing

Topological fields are sequences of one or more contiguous phrases found in an enclosing syntactic region, which is the clause in the case of the German topological field model (Höhle, 1983). These fields may have constraints on the number of words or phrases they contain, and do not necessarily form a semantically coherent constituent. In German, the topology serves to identify all of the components of the verbal head of a clause, as well as clause-level structure such as complementizers and subordinating conjunctions. Topological fields are a useful abstraction of word order, because while Germanic word order is relatively free with respect to grammatical functions, the order of the topological fields is strict and unvarying.

A German clause can be considered to be anchored by two "brackets" which contain modals, verbs and complementizers. The left bracket (*linke Klammer*, **LK**) may contain a complementizer, subordinating conjunction, or a finite verb, depending on the clause type, and the right bracket contains the verbal complex (**VC**). The other topological fields are defined in relation to these two brackets, and contain all other parts of the clause such as verbal arguments, adjuncts, and discourse cues.

The **VF** (*Vorfeld* or "pre-field") is so-named because it occurs before the left bracket. As the first constituent of most matrix clauses in declarative sentences, it has special significance for the coherence of a passage, which we will further discuss below. The **MF** (*Mittelfeld* or "middle field") is the field bounded by the two brackets. Most verb arguments, adverbs, and prepositional phrases are found here, unless they have been fronted and put in the VF, or are prosodically heavy and postposed to the NF field. The **NF** (*Nachfeld* or "post-field") contains prosodically heavy elements such as postposed prepositional phrases or relative clauses, and occasionally postposed noun phrases.

2.2 The Role of the Vorfeld

One of the reasons that we use topological fields for local coherence modelling is the role that the VF plays in signalling the information structure of German clauses, as it often contains the topic of the sentence.

In fact, its role is much more complex than being simply the topic position. Dipper and Zinsmeister (2009) distinguish multiple uses of the VF depending on whether it contains an element related to the surrounding discourse. They find that 45.1% of VFs are clearly related to the previous context by a reference or discourse relation, and a further 21.9% are deictic and refer to the situation described in the passage in a corpus study. They also run a sentence insertion experiment where subjects are asked to place an extracted sentence in its original location in a passage. The authors remark that extracted sentences with VFs that are referentially related to previous context (e.g., they contain a coreferential noun phrase or a discourse relation like "therefore") are reinserted at higher accuracies.

a)

#	**Original Sentence and Translation**
1	*Einen Zufluchtsort für Frauen, die von ihren Männern mißhandelt werden, gibt es nunmehr auch in Treptow.* "There is now a sanctuary for women who are mistreated by their husbands in Treptow as well."
2	*Das Bezirksamt bietet Frauen (auch mit Kindern) in derartigen Notsituationen vorübergehend eine Unterkunft.* "The district office offers women (even with children) in this type of emergency temporary accommodation."
3	*Zugleich werden die Betroffenen der Regelung des Unterhalts, bei Behördengängen und auch bei der Wohnungssuche unterstützt.* "At the same time, the affected are supported with provisions of necessities, in dealing with authorities, and also in the search for new accommodations."

b)

DE EN	*Zufluchtsort* sanctuary	*Frauen* women	*Männern* husbands	*Treptow* Treptow	*Kindern* children
1	acc	oth	oth	oth	–
2	–	oth	–	–	oth
3	–	nom	–	–	–

c)

– – 0.3	– nom 0.0	– acc 0.0	– oth 0.1	nom – 0.0	nom nom 0.0	nom acc 0.0	nom oth 0.0
acc – 0.1	acc nom 0.0	acc acc 0.0	acc oth 0.0	oth – 0.3	oth nom 0.1	oth acc 0.0	oth oth 0.1

Table 1: a) An example of a document from TüBa-D/Z, b) an abbreviated entity grid representation of it, and c) the feature vector representation of the abbreviated entity grid for transitions of length two. Mentions of the entity *Frauen* are underlined. nom: nominative, acc: accusative, oth: dative, oblique, and other arguments

Filippova and Strube (2007c) also examine the role of the VF in local coherence and natural language generation, focusing on the correlation between VFs and sentential topics. They follow Jacobs (2001) in distinguishing the *topic of addressation*, which is the constituent for which the proposition holds, and *frame-setting topics*, which is the domain in which the proposition holds, such as a temporal expression. They show in a user study that frame-setting topics are preferred to topics of addressation in the VF, except when a constituent needs to be established as the topic of addressation.

2.3 Using Entity Grids to Model Local Coherence

Barzilay and Lapata (2008) introduce the entity grid as a method of representing the coherence of a document. Entity grids indicate the location of the occurrences of an entity in a document, which is important for coherence modelling because mentions of an entity tend to appear in clusters of neighbouring or nearby sentences in coherent documents. This last assumption is adapted from Centering Theory approaches to discourse modelling.

In Barzilay and Lapata (2008), an entity grid is constructed for each document, and is represented as a matrix in which each row represents a sentence, and each column represents an entity. Thus, a cell in the matrix contains information about an entity in a sentence. The cell is marked by the presence or absence of the entity, and can also be augmented with other information about the entity in this sentence, such as the grammatical role of the noun phrase representing that entity in that sentence, or the topological field in which the noun phrase appears.

Consider the document in Table 1. An entity grid representation which incorporates the syntactic role of the noun phrase in which the entity ap-

pears is also shown (not all entities are listed for brevity). We tabulate the transitions of entities between different syntactic positions (or their non-occurrence) in sentences, and convert the frequencies of transitions into a feature vector representation of transition probabilities in the document.

To calculate transition probabilities, we divide the frequency of a particular transition by the total number of transitions of that length.

This model of local coherence was investigated for German by Filippova and Strube (2007a). The main focus of that work, however, was to adapt the model for use in a low-resource situation when perfect coreference information is not available. This is particularly useful in natural language understanding tasks. They employ a semantic clustering model to relate entities. In contrast, our work focuses on improving performance by annotating entities with additional linguistic information, such as topological fields, and is geared towards natural language generation systems where perfect information is available.

Similar models of local coherence include various Centering Theory accounts of local coherence ((Kibble and Power, 2004; Poesio et al., 2004) *inter alia*). The model of Elsner and Charniak (2007) uses syntactic cues to model the discourse-newness of noun phrases. There are also more global content models of topic shifts between sentences like Barzilay and Lee (2004).

3 Sentence Ordering Experiments

3.1 Method

We test a version of the entity grid representation augmented with topological fields in a sentence ordering experiment corresponding to Experiment 1 of Barzilay and Lapata (2008). The task is a binary classification task to identify the original version of a document from another version which contains the sentences in a randomly permuted order, which is taken to be incoherent. We solve this problem in a supervised machine learning setting, where the input is the feature vector representations of the two versions of the document, and the output is a binary value indicating the document with the original sentence ordering. We use `SVMlight`'s ranking module for classification (Joachims, 2002).

The corpus in our experiments consists of the last 480 documents of TüBa-D/Z version 4 (Telljohann et al., 2004), which contains manual coreference, grammatical role and topological field information. This set is larger than the set that was used in Experiment 1 of Barzilay and Lapata (2008), which consists of 400 documents in two English subcorpora on earthquakes and accidents respectively. The average document length in the TüBa-D/Z subcorpus is also greater, at 19.2 sentences compared to about 11 for the two subcorpora. Up to 20 random permutations of sentences were generated from each document, with duplicates removed.

There are 216 documents and 4126 original-permutation pairs in the training set, and 24 documents and 465 pairs in the development set. The remaining 240 documents are in the final test set (4243 pairs). The entity-based model is parameterized as follows.

Transition length – the maximum length of the transitions used in the feature vector representation of a document.

Representation – when marking the presence of an entity in a sentence, what information about the entity is marked (topological field, grammatical role, or none). We will describe the representations that we try in section 3.2.

Salience – whether to set a threshold for the frequency of occurrence of entities. If this is set, all entities below a certain frequency are treated separately from those reaching this frequency threshold when calculating transition probabilities. In the example in Table 1, with a salience threshold of 2, *Frauen* would be treated separately from *Männern* or *Kindern*.

Transition length, salience, and a regularization parameter are tuned on the development set. We only report results using the setting of transition length ≤ 4, and no salience threshold, because they give the best performance on the development set. This is in contrast to the findings of Barzilay and Lapata (2008), who report that transition length ≤ 3 and a salience threshold of 2 perform best on their data.

3.2 Entity Representations

The main goal of this study is to compare word order, grammatical role and topological field information, which is encoded into the entity grid at each occurrence of an entity. Here, we describe the variants of the entity representations that we compare.

Baseline Representations We implement several baseline representations against which we test our topological field-enhanced model. The simplest baseline representation marks the mere appearance of an entity without any additional information, which we refer to as `default`.

Another class of baseline representations mark the order in which entities appear in the clause. The correlation between word order and information structure is well known, and has formed the basis of some theories of syntax such as the Prague School's (Sgall et al., 1986). The two versions of clausal order we tried are `order 1/2/3+`, which marks a noun phrase as the first, the second, or the third or later to appear in a clause, and `order 1/2+`, which marks a noun phrase as the first, or the second or later to appear in a clause. Since noun phrases can be embedded in other noun phrases, overlaps can occur. In this case, the dominating noun phrase takes the smallest order number among its dominated noun phrases.

The third class of baseline representations we employ mark an entity by its grammatical role in the clause. Barzilay and Lapata (2008) found that grammatical role improves performance in this task for an English corpus. Because German distinguishes more grammatical roles morphologically than English, we experiment with various granularities of role labelling. In particular, `subj/obj` distinguishes the subject position, the object position, and another category for all other positions. `cases` distinguishes five types of entities corresponding to the four morphological cases of German in addition to another category for noun phrases which are not complements of the main verb.

Topological Field-Based These representations mark the topological field in which an entity appears. Some versions mark entities which are prepositional objects separately. We try versions which distinguish VF from non-VF, as well as more general versions that make use of a greater set of topological fields. `vf` marks the noun phrase as belonging to a VF (and not in a PP) or not. `vfpp` is the same as above, but allows prepositional objects inside the VF to be marked as VF. `topf/pp` distinguishes entities in the topological fields VF, MF, and NF, contains a separate category for PP, and a category for all other noun phrases. `topf` distinguishes between VF, MF, and NF, on the one hand, and everything else on the other. Prepositional objects are treated the same as other noun phrases here.

Combined We tried a representation which combines grammatical role and topological field into a single representation, `subj/obj`$\times$`vf`, which takes the Cartesian product of `subj/obj` and `vf` above.

Topological fields do not map directly to topic-focus distinctions. For example, besides the topic of the sentence, the Vorfeld may contain discourse cues, expletive pronouns, or the informational or contrastive focus. Furthermore, there are additional constraints on constituent order related to pronominalization. Thus, we devised additional entity representations to account for these aspects of German.

`topic` attempts to identify the sentential topic of a clause. A noun phrase is marked as TOPIC if it is in VF as in `vfpp`, or if it is the first noun phrase in MF and also the first NP in the clause. Other noun phrases in MF are marked as NONTOPIC. Categories for NF and miscellaneous noun phrases also exist. While this representation may appear to be very similar to simply distinguishing the first entity in a clause as for `order 1/2+` in that TOPIC would correspond to the first entity in the clause, they are in fact distinct. Due to issues related to coordination, appositive constructions, and fragments which do not receive a topology of fields, the first entity in a clause is labelled the TOPIC only 80.8% of the time in the corpus. This representation also distinguishes NFs, which clausal order does not.

`topic+pron` refines the above by taking into account a word order restriction in German that pronouns appear before full noun phrases in the MF field. The following set of decisions represents how a noun phrase is marked: If the first NP in the clause is a pronoun in an MF field and is the subject, we mark it as TOPIC. If it is not the subject, we mark it as NONTOPIC. For other NPs, we follow the `topic` representation.

3.3 Automatic annotations

While it is reasonable to assume perfect annotations of topological fields and grammatical roles in many NLG contexts, this assumption may be less appropriate in other applications involving text-to-text generation where the input to the system is text such as paraphrasing or machine translation. Thus, we test the robustness of the entity repre-

Representation	*Manual*	*Automatic*
`topf/pp`	**94.44**	**94.89**
`topic`	94.13	94.53
`topic+pron`	94.08	94.51
`topf`	93.87	93.11
`subj/obj`	93.83[1]	91.7++
`cases`	93.31[2]	90.93++
`order 1/2+`	92.51++	92.1+
`subj/obj`×`vf`	92.32++	90.74++
`default`	91.42++	91.42++
`vfpp`	91.37++	91.68++
`vf`	91.21++	91.16++
`order 1/2/3+`	91.16++	90.71++

Table 2: Accuracy (%) of the permutation detection experiment with various entity representations using manual and automatic annotations of topological fields and grammatical roles. The baseline without any additional annotation is underlined. Two-tailed sign tests were calculated for each result against the best performing model in each column ([1]: $p = 0.101$; [2]: $p = 0.053$; +: statistically significant, $p < 0.05$; ++: very statistically significant, $p < 0.01$).

sentations to automatic extraction in the absence of manual annotations. We employ the following two systems for extracting topological fields and grammatical roles.

To parse topological fields, we use the Berkeley parser of Petrov and Klein (2007), which has been shown to perform well at this task (Cheung and Penn, 2009). The parser is trained on sections of TüBa-D/Z which do not overlap with the section from which the documents for this experiment were drawn, and obtains an overall parsing performance of 93.35% F_1 on topological fields and clausal nodes without gold POS tags on the section of TüBa-D/Z it was tested on.

We tried two methods to obtain grammatical roles. First, we tried extracting grammatical roles from the parse trees which we obtained from the Berkeley parser, as this information is present in the edge labels that can be recovered from the parse. However, we found that we achieved better accuracy by using `RFTagger` (Schmid and Laws, 2008), which tags nouns with their morphological case. Morphological case is distinct from grammatical role, as noun phrases can function as adjuncts in possessive constructions and prepositional phrases. However, we can approximate the grammatical role of an entity using the morphological case. We follow the annotation conventions of TüBa-D/Z in not assigning a grammatical role when the noun phrase is a prepositional object. We also do not assign a grammatical role when the noun phrase is in the genitive case, as genitive objects are very rare in German and are far outnumbered by the possessive genitive construction.

Annotation	*Accuracy* (%)
Grammatical role	83.6
Topological field (+PP)	93.8
Topological field (−PP)	95.7
Clausal order	90.8

Table 3: Accuracy of automatic annotations of noun phrases with coreferents. +PP means that prepositional objects are treated as a separate category from topological fields. −PP means they are treated as other noun phrases.

3.4 Results

Table 2 shows the results of the sentence ordering permutation detection experiment. The top four performing entity representations are all topological field-based, and they outperform grammatical role-based and simple clausal order-based models. These results indicate that the information that topological fields provide about clause structure, appositives, right dislocation, etc. which is not captured by simple clausal order is important for coherence modelling. The representations incorporating linguistics-based heuristics do not outperform purely topological field-based models.

Surprisingly, the VF-based models fare quite poorly, performing worse than not adding any annotations, despite the fact that topological field-based models in general perform well. This result may be a result of the heterogeneous uses of the VF.

The automatic topological field annotations are more accurate than the automatic grammatical role annotations (Table 3), which may partly explain why grammatical role-based models suffer more when using automatic annotations. Note, however, that the models based on automatic topological field annotations outperform even the grammatical role-based models using manual annotation (at marginal significance, $p < 0.1$). The topo-

logical field annotations are accurate enough that automatic annotations produce no decrease in performance.

These results show the upper bound of entity-based local coherence modelling with perfect coreference information. The results we obtain are higher than the results for the English corpora of Barzilay and Lapata (2008) (87.2% on the Earthquakes corpus and 90.4% on the Accidents corpus), but this is probably due to corpus differences as well as the availability of perfect coreference information in our experiments[1].

Due to the high performance we obtained, we calculated Kendall tau coefficients (Lapata, 2006) over the sentence orderings of the cases in which our best performing model is incorrect, to determine whether the remaining errors are instances where the permuted ordering is nearly identical to the original ordering. We obtained a τ of 0.0456 in these cases, compared to a τ of -0.0084 for all the pairs, indicating that this is not the case.

To facilitate comparison to the results of Filippova and Strube (2007a), we rerun this experiment on the same subsections of the corpus as in that work for training and testing. The first 100 articles of TüBa-D/Z are used for testing, while the next 200 are used for training and development.

Unlike the previous experiments, we do not do parameter tuning on this set of data. Instead, we follow Filippova and Strube (2007a) in using transition lengths of up to three. We do not put in a salience threshold. We see that our results are much better than the ones reported in that work, even for the `default` representation. The main reason for this discrepancy is probably the way that entities are created from the corpus. In our experiments, we create an entity for every single noun phrase node that we encounter, then merge the entities that are linked by coreference. Filippova and Strube (2007a) convert the annotations of TüBa-D/Z into a dependency format, then extract entities from the noun phrases found there. They may thus annotate fewer entities, as there may be nested NP nodes in the original corpus. There may also be noise in the dependency conversion process.

[1]Barzilay and Lapata (2008) use the coreference system of Ng and Cardie (2002) to obtain coreference annotations. We are not aware of similarly well-tested, publicly available coreference resolution systems that handle all types of anaphora for German. We considered adapting the BART coreference resolution toolkit (Versley et al., 2008) to German, but a number of language-dependent decisions regarding preprocessing, feature engineering, and the learning paradigm would need to be made in order to achieve reasonable performance comparable to state-of-the-art English coreference resolution systems.

Representation	*Accuracy* (%)
`topf/pp`	**93.83**
`topic`	93.31
`topic+pron`	93.31
`topf`	92.49
`subj/obj`	88.99
`order 1/2+`	88.89
`order 1/2/3+`	88.84
`cases`	88.63
`vf`	87.60
`vfpp`	88.17
`default`	87.55
`subj/obj`$\times$`vf`	87.71
(Filippova and Strube, 2007)	75

Table 4: Accuracy (%) of permutation detection experiment with various entity representations using manual and automatic annotations of topological fields and grammatical roles on subset of corpus used by Filippova and Strube (2007a).

The relative rankings of different entity representations in this experiment are similar to the rankings of the previous experiment, with topological field-based models outperforming grammatical role and clausal order models.

4 Local Coherence for Natural Language Generation

One of the motivations of the entity grid-based model is to improve surface realization decisions in NLG systems. A typical experimental design would pass the contents of the test section of a corpus as input to the NLG system with the ordering information stripped away. The task is then to regenerate the ordering of the information found in the original corpus. Various coherence models have been tested in corpus-based NLG settings. For example, Karamanis et al. (2009) compare several versions of Centering Theory-based metrics of coherence on corpora by examining how highly the original ordering found in the corpus is ranked compared to other possible orderings of propositions. A metric performs well if it ranks the original ordering better than the alternative orderings.

In our next experiment, we incorporate local co-

herence information into the system of Filippova and Strube (2007b). We embed entity topological field transitions into their probabilistic model, and show that the added coherence component slightly improves the performance of the baseline NLG system in generating constituent orderings in a German corpus, though not to a statistically significant degree.

4.1 Method

We use the WikiBiography corpus[2] for our experiments. The corpus consists of more than 1100 biographies taken from the German Wikipedia, and contains automatic annotations of morphological, syntactic, and semantic information. Each article also contains the coreference chain of the subject of the biography (the biographee). The first 100 articles are used for testing, the next 200 for development, and the rest for training.

The baseline generation system already incorporates topological field information into the constituent ordering process. The system operates in two steps. First, in main clauses, one constituent is selected as the Vorfeld (VF). This is done using a maximum entropy model (call it `MAXENT`). Then, the remaining constituents are ordered using a second maximum entropy model (`MAXENT2`). Significantly, Filippova and Strube (2007b) found that selecting the VF first, and then ordering the remaining constituents results in a 9% absolute improvement over the corresponding model where the selection is performed in one step by the sorting algorithm alone.

The maximum entropy model for both steps rely on the following features:

- features on the voice, valency, and identity of the main verb of the clause
- features on the morphological and syntactic status of the constituent to be ordered
- whether the constituent occurs in the preceding sentence
- features for whether the constituent contains a determiner, an anaphoric pronoun, or a relative clause
- the size of the constituent in number of modifiers, in depth, and in number of words
- the semantic class of the constituent (person, temporal, location, etc.) The biographee, in particular, is marked by its own semantic class.

In the first VF selection step, `MAXENT` simply produces a probability of each constituent being a VF, and the constituent with the highest probability is selected. In the second step, `MAXENT2` takes the featural representation of two constituents, and produces an output probability of the first constituent preceding the second constituent. The final ordering is achieved by first randomizing the order of the constituents in a clause (besides the first one, which is selected to be the VF), then sorting them according to the precedence probabilities. Specifically, a constituent A is put before a constituent B if `MAXENT2`(A,B) > 0.5. Because this precedence relation is not antisymmetric (i.e., `MAXENT2`(A,B) > 0.5 and `MAXENT2`(B,A) > 0.5 may be simultaneously true or simultaneously false), different initializations of the order produce different sorted results. In our experiments, we correct this by defining the precedence relation to be A precedes B iff `MAXENT2`(A,B) $>$ `MAXENT2`(B,A). This change does not greatly impact the performance, and removes the randomized element of the algorithm.

The baseline system does not directly model the context when ordering constituents. All of the features but one in the original maximum entropy models rely on local properties of the clause. We incorporate local coherence information into the model by adding entity transition features which we found to be useful in the sentence ordering experiment in Section 3 above.

Specifically, we add features indicating the topological fields in which entities occur in the previous sentences. We found that looking back up to two sentences produces the best results (by tuning on the development set). Because this corpus does not come with general coreference information except for the coreference chain of the biographee, we use the semantic classes instead. So, all constituents in the same semantic class are treated as one coreference chain. An example of a feature may be **biog-last2**, which takes on a value such as 'v−', meaning that this constituent refers to the biographee, and the biographee occurs in the VF two clauses ago (v), but does not appear in the previous clause (−). For a constituent which is not the biographee, this feature would be marked

[2] `http://www.eml-research.de/english/research/nlp/download/wikibiography.php`

Method	*VF Acc* (%)	*Acc* (%)	*Tau*
Baseline	68.7	60.9	**0.72**
+Coherence	**69.2**	**61.5**	**0.72**

Table 5: Results of adding coherence features into a natural language generation system. VF Acc% is the accuracy of selecting the first constituent in main clauses. Acc % is the percentage of perfectly ordered clauses, tau is Kendall's τ on the constituent ordering. The test set contains 2246 clauses, of which 1662 are main clauses.

'na' (not applicable).

4.2 Results

Table 5 shows the results of adding these contextual features into the maximum entropy models. We see that we obtain a small improvement in the accuracy of VF selection, and in the accuracy of correctly ordering the entire clause. These improvements are not statistically significant by McNemar's test. We suggest that the lack of coreference information for all entities in the article may have reduced the benefit of the coherence component. Also, the topline of performance is substantially lower than 100%, as multiple orderings are possible and equally valid. Human judgements on information structuring for both inter- and intra-sentential units are known to have low agreement (Barzilay et al., 2002; Filippova and Strube, 2007c; Lapata, 2003; Chen et al., 2007). Thus, the relative error reduction is higher than the absolute reduction might suggest.

5 Conclusions

We have shown that topological fields are a useful source of information for local coherence modelling. In a sentence-order permutation detection task, models which use topological field information outperform both grammatical role-based models and models based on simple clausal order, with the best performing model achieving a relative error reduction of 40.4% over the original baseline without any additional annotation. Applying our local coherence model in another setting, we have embedded topological field transitions of entities into an NLG system which orders constituents in German clauses. We find that the coherence-enhanced model slightly outperforms the baseline system, but this was not statistically significant.

We suggest that the utility of topological fields in local coherence modelling comes from the interaction between word order and information structure in freer-word-order languages. Crucially, topological fields take into account issues such as coordination, appositives, sentential fragments and differences in clause types, which word order alone does not. They are also shallow enough to be accurately parsed automatically for use in resource-poor applications.

Further refinement of the topological field annotations to take advantage of the fact that they do not correspond neatly to any single information status such as topic or focus could provide additional performance gains. The model also shows promise for other discourse-related tasks such as coreference resolution and discourse parsing.

Acknowledgements

We are grateful to Katja Filippova for providing us with source code for the experiments in Section 4 and for answering related questions, and to Timothy Fowler for useful discussions and comments on a draft of the paper. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proc. HLT-NAACL 2004*, pages 113–120.

R. Barzilay, N. Elhadad, and K. McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

E. Chen, B. Snyder, and R. Barzilay. 2007. Incremental text structuring with online hierarchical ranking. In *Proceedings of EMNLP*, pages 83–91.

J.C.K. Cheung and G. Penn. 2009. Topological Field Parsing of German. In *Proc. 47th ACL and 4th IJCNLP*, pages 64–72. Association for Computational Linguistics.

S. Dipper and H. Zinsmeister. 2009. The Role of the German Vorfeld for Local Coherence: A Pilot Study. In *Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 69–79. Gunter Narr.

M. Elsner and E. Charniak. 2007. A generative discourse-new model for text coherence. Technical report, Technical Report CS-07-04, Brown University.

K. Filippova and M. Strube. 2007a. Extending the entity-grid coherence model to semantically related entities. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, pages 139–142. Association for Computational Linguistics.

K. Filippova and M. Strube. 2007b. Generating constituent order in German clauses. In *Proc. 45th ACL*, pages 320–327.

K. Filippova and M. Strube. 2007c. The German Vorfeld and Local Coherence. *Journal of Logic, Language and Information*, 16(4):465–485.

T.N. Höhle. 1983. *Topologische Felder*. Ph.D. thesis, Köln.

J. Jacobs. 2001. The dimensions of topiccomment. *Linguistics*, 39(4):641–681.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines*. Kluwer.

N. Karamanis, C. Mellish, M. Poesio, and J. Oberlander. 2009. Evaluating centering for information ordering using corpora. *Computational Linguistics*, 35(1):29–46.

R. Kibble and R. Power. 2004. Optimizing referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

M. Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proc. 41st ACL*, pages 545–552.

M. Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proc. 40th ACL*, pages 104–111.

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

M. Poesio, R. Stevenson, B.D. Eugenio, and J. Hitzeman. 2004. Centering: A parametric theory and its instantiations. *Computational Linguistics*, 30(3):309–363.

H. Schmid and F. Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proc. 22nd COLING*, pages 777–784. Association for Computational Linguistics.

P. Sgall, E. Hajičová, J. Panevová, and J. Mey. 1986. *The meaning of the sentence in its semantic and pragmatic aspects*. Springer.

M. Strube and U. Hahn. 1999. Functional centering: Grounding referential coherence in information structure. *Computational Linguistics*, 25(3):309–344.

H. Telljohann, E. Hinrichs, and S. Kubler. 2004. The TüBa-D/Z treebank: Annotating German with a context-free backbone. In *Proc. Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2229–2235.

Y. Versley, S.P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proc. 46th ACL-HLT Demo Session*, pages 9–12. Association for Computational Linguistics.

Syntactic and Semantic Factors in Processing Difficulty: An Integrated Measure

Jeff Mitchell, Mirella Lapata, Vera Demberg and Frank Keller
University of Edinburgh
Edinburgh, United Kingdom
jeff.mitchell@ed.ac.uk, mlap@inf.ed.ac.uk,
v.demberg@ed.ac.uk, keller@inf.ed.ac.uk

Abstract

The analysis of reading times can provide insights into the processes that underlie language comprehension, with longer reading times indicating greater cognitive load. There is evidence that the language processor is highly predictive, such that prior context allows upcoming linguistic material to be anticipated. Previous work has investigated the contributions of semantic and syntactic contexts in isolation, essentially treating them as independent factors. In this paper we analyze reading times in terms of a single predictive measure which integrates a model of semantic composition with an incremental parser and a language model.

1 Introduction

Psycholinguists have long realized that language comprehension is highly *incremental*, with readers and listeners continuously extracting the meaning of utterances on a word-by-word basis. As soon as they encounter a word in a sentence, they integrate it as fully as possible into a representation of the sentence thus far (Marslen-Wilson 1973; Konieczny 2000; Tanenhaus et al. 1995; Sturt and Lombardo 2005). Recent research suggests that language comprehension can also be highly *predictive*, i.e., comprehenders are able to anticipate upcoming linguistic material. This is beneficial as it gives them more time to keep up with the input, and predictions can be used to compensate for problems with noise or ambiguity.

Two types of prediction have been observed in the literature. The first type is semantic prediction, as evidenced in semantic priming: a word that is preceded by a semantically related prime or a semantically congruous sentence fragment is processed faster (Stanovich and West 1981; van Berkum et al. 1999; Clifton et al. 2007). Another example is argument prediction: listeners are able to launch eye-movements to the predicted argument of a verb before having encountered it, e.g., they will fixate an edible object as soon as they hear the word *eat* (Altmann and Kamide 1999). The second type of prediction is syntactic prediction. Comprehenders are faster at naming words that are syntactically compatible with prior context, even when they bear no semantic relationship to the context (Wright and Garrett 1984). Another instance of syntactic prediction has been reported by Staub and Clifton (2006): following the word *either*, readers predict *or* and the complement that follows it, and process it faster compared to a control condition without *either*.

Thus, human language processing takes advantage of the constraints imposed by the preceding semantic and syntactic context to derive expectations about the upcoming input. Much recent work has focused on developing computational measures of these constraints and expectations. Again, the literature is split into syntactic and semantic models. Probably the best known measure of syntactic expectation is *surprisal* (Hale 2001) which can be coarsely defined as the negative log probability of word w_t given the preceding words, typically computed using a probabilistic context-free grammar.

Modeling work on semantic constraint focuses on the degree to which a word is related to its preceding context. Pynte et al. (2008) use Latent Semantic Analysis (LSA, Landauer and Dumais 1997) to assess the degree of contextual constraint exerted on a word by its context. In this framework, word meanings are represented as vectors in a high dimensional space and distance in this space is interpreted as an index of processing difficulty. Other work (McDonald and Brew 2004) models contextual constraint in information theoretic terms. The assumption is that words carry prior *semantic expectations* which are updated upon seeing the next word. Expectations are represented by a vector of probabilities which reflects the likely location in semantic space of the upcoming word.

The measures discussed above are typically computed automatically on real-language corpora using data-driven methods and their predictions are verified through analysis of eye-movements that people make while reading. Ample evidence

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 196–206,
Uppsala, Sweden, 11-16 July 2010.

(Rayner 1998) demonstrates that eye-movements are related to the moment-to-moment cognitive activities of readers. They also provide an accurate temporal record of the on-line processing of natural language, and through the analysis of eye-movement measurements (e.g., the amount of time spent looking at a word) can give insight into the processing difficulty involved in reading.

In this paper, we investigate a model of prediction that is incremental and takes into account syntactic as well as semantic constraint. The model essentially integrates the predictions of an incremental parser (Roark 2001) together with those of a semantic space model (Mitchell and Lapata 2009). The latter creates meaning representations *compositionally*, and therefore builds semantic expectations for word sequences (e.g., phrases, sentences, even documents) rather than isolated words. Some existing models of sentence processing integrate semantic information into a probabilistic parser (Narayanan and Jurafsky 2002; Padó et al. 2009); however, the semantic component of these models is limited to semantic role information, rather than attempting to build a full semantic representation for a sentence. Furthermore, the models of Narayanan and Jurafsky (2002) and Padó et al. (2009) do not explicitly model prediction, but rather focus on accounting for garden path effects. The proposed model simultaneously captures semantic and syntactic effects in a single measure which we empirically show is predictive of processing difficulty as manifested in eye-movements.

2 Models of Processing Difficulty

As described in Section 1, reading times provide an insight into the various cognitive activities that contribute to the overall processing difficulty involved in comprehending a written text. To quantify and understand the overall cognitive load associated with processing a word in context, we will break that load down into a sum of terms representing distinct computational costs (semantic and syntactic). For example, surprisal can be thought of as measuring the cost of dealing with unexpected input. When a word conforms to the language processor's expectations, surprisal is low, and the cognitive load associated with processing that input will also be low. In contrast, unexpected words will have a high surprisal and a high cognitive cost.

However, high-level syntactic and semantic factors are only one source of cognitive costs. A sizable proportion of the variance in reading times is accounted for by costs associated with low-level features of the stimuli, e.g.. relating to orthography and eye-movement control (Rayner 1998). In addition, there may also be costs associated with the integration of new input into an incremental representation. Dependency Locality Theory (DLT, Gibson 2000) is essentially a distance-based measure of the amount of processing effort required when the head of a phrase is integrated with its syntactic dependents. We do not consider integration costs here (as they have not been shown to correlate reliably with reading times; see Demberg and Keller 2008 for details) and instead focus on the costs associated with semantic and syntactic constraint and low-level features, which appear to make the most substantial contributions.

In the following subsections we describe the various features which contribute to the processing costs of a word in context. We begin by looking at the low-level costs and move on to consider the costs associated with syntactic and semantic constraint. For readers unfamiliar with the methodology involved in modeling eye-tracking data, we note that regression analysis (or the more general mixed effects models) is typically used to study the relationship between *dependent* and *independent* variables. The independent variables are the various costs of processing effort and the dependent variables are measurements of eye-movements, three of which are routinely used in the literature: *first fixation duration* (the duration of the first fixation on a word regardless of whether it is the first fixation on a word or the first of multiple fixations on the same word), *first pass duration*, also known as *gaze duration*, (the sum of all fixations made on a word prior to looking at another word), and *total reading time* (the sum of all fixations on a word including refixations after moving on to other words).

2.1 Low-level Costs

Low-level features include word frequency (more frequent words are read faster), word length (shorter words are read faster), and the position of the word in the sentence (later words are read faster). Oculomotor variables have also been found to influence reading times. These include previous fixation (indicating whether or not the previous word has been fixated), launch distance (how many characters intervene between the current fixation and the previous fixation), and landing position (which letter in the word the fixation landed on).

Information about the sequential context of a word can also influence reading times. Mc-

Donald and Shillcock (2003) show that forward and backward transitional probabilities are predictive of first fixation and first pass durations: the higher the transitional probability, the shorter the fixation time. Backward transitional probability is essentially the conditional probability of a word given its immediately preceding word, $P(w_k|w_{k-1})$. Analogously, forward probability is the conditional probability of the current word given the next word, $P(w_k|w_{k+1})$.

2.2 Syntactic Constraint

As mentioned earlier, surprisal (Hale 2001; Levy 2008) is one of the best known models of processing difficulty associated with syntactic constraint, and has been previously applied to the modeling of reading times (Demberg and Keller 2008; Ferrara Boston et al. 2008; Roark et al. 2009; Frank 2009). The basic idea is that the processing costs relating to the expectations of the language processor can be expressed in terms of the probabilities assigned by some form of language model to the input. These processing costs are assumed to arise from the change in the expectations of the language processor as new input arrives. If we express these expectations in terms of a distribution over all possible continuations of the input seen so far, then we can measure the magnitude of this change in terms of the Kullback-Leibler divergence of the old distribution to the updated distribution. This measure of processing cost for an input word, w_{k+1}, given the previous context, $w_1 \dots w_k$, can be expressed straightforwardly in terms of its conditional probability as:

$$S = -\log P(w_{k+1}|w_1 \dots w_k) \qquad (1)$$

That is, the processing cost for a word decreases as its probability increases, with zero processing cost incurred for words which must appear in a given context, as these do not result in any change in the expectations of the language processor.

The original formulation of surprisal (Hale 2001) used a probabilistic parser to calculate these probabilities, as the emphasis was on the processing costs incurred when parsing structurally ambiguous garden path sentences.[1] Several variants of calculating surprisal have been developed in the literature since using different parsing strategies (e.g., left-to-right vs. top-down, PCFGs vs dependency parsing) and different degrees of lexicalization (see Roark et al. 2009 for an overview) . For instance, unlexicalized surprisal can be easily derived by substituting the words in Equation (1) with parts of speech (Demberg and Keller 2008). Surprisal could be also defined using a vanilla language model that does not take any structural or grammatical information into account (Frank 2009).

2.3 Semantic Constraint

Distributional models of meaning have been commonly used to quantify the semantic relation between a word and its context in computational studies of lexical processing. These models are based on the idea that words with similar meanings will be found in similar contexts. In putting this idea into practice, the meaning of a word is then represented as a vector in a high dimensional space, with the vector components relating to the strength on occurrence of that word in various types of context. Semantic similarities are then modeled in terms of geometric similarities within the space.

To give a concrete example, Latent Semantic Analysis (LSA, Landauer and Dumais 1997) creates a meaning representation for words by constructing a word-document co-occurrence matrix from a large collection of documents. Each row in the matrix represents a word, each column a document, and each entry the frequency with which the word appeared within that document. Because this matrix tends to be quite large it is often transformed via a singular value decomposition (Berry et al. 1995) into three component matrices: a matrix of word vectors, a matrix of document vectors, and a diagonal matrix containing singular values. Re-multiplying these matrices together using only the initial portions of each (corresponding to the use of a lower dimensional spatial representation) produces a tractable approximation to the original matrix. In this framework, the similarity between two words can be easily quantified, e.g., by measuring the cosine of the angle of the vectors representing them.

As LSA is one the best known semantic space models it comes as no surprise that it has been used to analyze semantic constraint. Pynte et al. (2008) measure the similarity between the next word and its preceding context under the assumption that high similarity indicates high semantic constraint (i.e., the word was expected) and analogously low similarity indicates low semantic constraint (i.e., the word was unexpected). They oper-

[1] While hearing a sentence like *The horse raced past the barn fell* (Bever 1970), English speakers are inclined to interpreted *horse* as the subject of *raced* expecting the sentence to end at the word *barn*. So upon hearing the word *fell* they are forced to revise their analysis of the sentence thus far and adopt a reduced relative reading.

ationalize preceding contexts in two ways, either as the word immediately preceding the next word as the sentence fragment preceding it. Sentence fragments are represented as the average of the words they contain independently of their order. The model takes into account only content words, function words are of little interest here as they can be found in any context.

Pynte et al. (2008) analyze reading times on the French part of the Dundee corpus (Kennedy and Pynte 2005) and find that word-level LSA similarities are predictive of first fixation and first pass durations, whereas sentence-level LSA is only predictive of first pass duration (i.e., for a measure that includes refixation). This latter finding is somewhat counterintuitive, one would expect longer contexts to have an immediate effect as they are presumably more constraining. One reason why sentence-level influences are only visible on first pass duration may be due to LSA itself, which is syntax-blind. Another reason relates to the way sentential context was modeled as vector addition (or averaging). The idea of averaging is not very attractive from a linguistic perspective as it blends the meanings of individual words together. Ideally, the combination of simple elements onto more complex ones must allow the construction of novel meanings which go beyond those of the individual elements (Pinker 1994).

The only other model of semantic constraint we are aware of is Incremental Contextual Distinctiveness (ICD, McDonald 2000; McDonald and Brew 2004). ICD assumes that words carry prior semantic expectations which are updated upon seeing the next word. Context is represented by a vector of probabilities which reflects the likely location in semantic space of the upcoming word. When the latter is observed, the prior expectation is updated using a Bayesian inference mechanism to reflect the newly arrived information. Like LSA, ICD is based on word co-occurrence vectors, however it does not employ singular value decomposition, and constructs a word-word rather than a word-document co-occurrence matrix. Although this model has been shown to successfully simulate single- and multiple-word priming (McDonald and Brew 2004), it failed to predict processing costs in the Embra eye-tracking corpus (McDonald and Shillcock 2003).

In this work we model semantic constraint using the representational framework put forward in Mitchell and Lapata (2008). Their aim is not so much to model processing difficulty, but to construct vector-based meaning representations that go beyond individual words. They introduce a general framework for studying vector *composition*, which they formulate as a function f of two vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{h} = f(\mathbf{u}, \mathbf{v}) \tag{2}$$

where $\mathbf{h}$ denotes the composition of $\mathbf{u}$ and $\mathbf{v}$. Different composition models arise, depending on how f is chosen. Assuming that $\mathbf{h}$ is a linear function of the Cartesian product of $\mathbf{u}$ and $\mathbf{v}$ allows to specify *additive* models which are by far the most common method of vector combination in the literature:

$$h_i = u_i + v_i \tag{3}$$

Alternatively, we can assume that $\mathbf{h}$ is a linear function of the tensor product of $\mathbf{u}$ and $\mathbf{v}$, and thus derive models based on *multiplication*:

$$h_i = u_i \cdot v_i \tag{4}$$

Mitchell and Lapata (2008) show that several additive and multiplicative models can be formulated under this framework, including the well-known tensor products (Smolensky 1990) and circular convolution (Plate 1995). Importantly, composition models are not defined with a specific semantic space in mind, they could easily be adapted to LSA, or simple co-occurrence vectors, or more sophisticated semantic representations (e.g., Griffiths et al. 2007), although admittedly some composition functions may be better suited for particular semantic spaces.

Composition models can be straightforwardly used as predictors of processing difficulty, again via measuring the cosine of the angle between a vector $\mathbf{w}$ representing the upcoming word and a vector $\mathbf{h}$ representing the words preceding it:

$$sim(\mathbf{w}, \mathbf{h}) = \frac{\mathbf{w} \cdot \mathbf{h}}{|\mathbf{w}||\mathbf{h}|} \tag{5}$$

where $\mathbf{h}$ is created compositionally, via some (additive or multiplicative) function f.

In this paper we evaluate additive and compositional models in their ability to capture semantic prediction. We also examine the influence of the underlying meaning representations by comparing a simple semantic space similar to McDonald (2000) against Latent Dirichlet Allocation (Blei et al. 2003; Griffiths et al. 2007). Specifically, the simpler space is based on word co-occurrence counts; it constructs the vector representing a given target word, t, by identifying all the tokens of t in a corpus and recording the counts of context words, c_i (within a specific window). The context words, c_i, are limited to a set of the n most

common content words and each vector component is given by the ratio of the probability of a c_i given t to the overall probability of c_i.

$$v_i = \frac{p(c_i|t)}{p(c_i)} \quad (6)$$

Despite its simplicity, the above semantic space (and variants thereof) has been used to successfully simulate lexical priming (e.g., McDonald 2000), human judgments of semantic similarity (Bullinaria and Levy 2007), and synonymy tests (Padó and Lapata 2007) such as those included in the Test of English as Foreign Language (TOEFL).

LDA is a probabilistic topic model offering an alternative to spatial semantic representations. It is similar in spirit to LSA, it also operates on a word-document co-occurrence matrix and derives a reduced dimensionality description of words and documents. Whereas in LSA words are represented as points in a multi-dimensional space, LDA represents words using topics. Specifically, each document in a corpus is modeled as a distribution over K topics, which are themselves characterized as distribution over words. The individual words in a document are generated by repeatedly sampling a topic according to the topic distribution and then sampling a single word from the chosen topic. Under this framework, word meaning is represented as a probability distribution over a set of latent topics, essentially a vector whose dimensions correspond to topics and values to the probability of the word given these topics. Topic models have been recently gaining ground as a more structured representation of word meaning (Griffiths et al. 2007; Steyvers and Griffiths 2007). In contrast to more standard semantic space models where word senses are conflated into a single representation, topics have an intuitive correspondence to coarse-grained sense distinctions.

3 Integrating Semantic Constraint into Surprisal

The treatment of semantic and syntactic constraint in models of processing difficulty has been somewhat inconsistent. While surprisal is a theoretically well-motivated measure, formalizing the idea of linguistic processing being highly predictive in terms of probabilistic language models, the measurement of semantic constraint in terms of vector similarities lacks a clear motivation. Moreover, the two approaches, surprisal and similarity, produce mathematically different types of measures. Formally, it would be preferable to have a single approach to capturing constraint and the obvious solution is to derive some form of semantic surprisal rather than sticking with similarity. This can be achieved by turning a vector model of semantic similarity into a probabilistic language model.

There are in fact a number of approaches to deriving language models from distributional models of semantics (e.g., Bellegarda 2000; Coccaro and Jurafsky 1998; Gildea and Hofmann 1999). We focus here on the model of Mitchell and Lapata (2009) which tackles the issue of the composition of semantic vectors and also integrates the output of an incremental parser. The core of their model is based on the product of a trigram model $p(w_n|w_{n-2}^{n-1})$ and a semantic component $\Delta(w_n,h)$ which determines the factor by which this probability should be scaled up or down given the prior semantic context h:

$$p(w_n) = p(w_n|w_{n-2}^{n-1}) \cdot \Delta(w_n,h) \quad (7)$$

The factor $\Delta(w_n,h)$ is essentially based on a comparison between the vector representing the current word w_n and the vector representing the prior history h. Varying the method for constructing word vectors (e.g., using LDA or a simpler semantic space model) and for combining them into a representation of the prior context h (e.g., using additive or multiplicative functions) produces distinct models of semantic composition.

The calculation of Δ is then based on a weighted dot product of the vector representing the upcoming word w, with the vector representing the prior context h:

$$\Delta(w,h) = \sum_i w_i h_i p(c_i) \quad (8)$$

As shown in Equation (7) this semantic factor then modulates the trigram probabilities, to take account of the effect of the semantic content outside the n-gram window.

Mitchell and Lapata (2009) show that a combined semantic-trigram language model derived from this approach and trained on the Wall Street Journal outperforms a baseline trigram model in terms of perplexity on a held out set. They also linearly interpolate this semantic language model with the output of an incremental parser, which computes the following probability:

$$p(w|h) = \lambda p_1(w|h) + (1-\lambda) p_2(w|h) \quad (9)$$

where $p_1(w|h)$ is computed as in Equation (7) and $p_2(w|h)$ is computed by the parser. Their implementation uses Roark's (2001) top-down incremental parser which estimates the probability of

the next word based upon the previous words of the sentence. These *prefix* probabilities are calculated from a grammar, by considering the likelihood of seeing the next word given the possible grammatical relations representing the prior context.

Equation (9) essentially defines a language model which combines semantic, syntactic and *n*-gram structure, and Mitchell and Lapata (2009) demonstrate that it improves further upon a semantic language model in terms of perplexity. We argue that the probabilities from this model give us a means to model the incrementally and predictivity of the language processor in a manner that integrates both syntactic and semantic constraints. Converting these probabilities to surprisal should result in a single measure of the processing cost associated with semantic and syntactic expectations.

4 Method

Data The models discussed in the previous section were evaluated against an eye-tracking corpus. Specifically, we used the English portion of the Dundee Corpus (Kennedy and Pynte 2005) which contains 20 texts taken from *The Independent* newspaper. The corpus consists of 51,502 tokens and 9,776 types in total. It is annotated with the eye-movement records of 10 English native speakers, who each read the whole corpus. The eye-tracking data was preprocessed following the methodology described in Demberg and Keller (2008). From this data, we computed total reading time for each word in the corpus. Our statistical analyses were based on actual reading times, and so we only included words that were not skipped. We also excluded words for which the previous word had been skipped, and words on which the normal left-to-right movement of gaze had been interrupted, i.e., by blinks, regressions, etc. Finally, because our focus is the influence of semantic context, we selected only content words whose prior sentential context contained at least two further content words. The resulting data set consisted of 53,704 data points, which is about 10% of the theoretically possible total.[2]

Model Implementation All elements of our model were trained on the BLLIP corpus, a collection of texts from the Wall Street Journal (years 1987–89). The training corpus consisted of 38,521,346 words. We used a development corpus of 50,006 words and a test corpus of similar size. All words were converted to lowercase and numbers were replaced with the symbol $\langle$num$\rangle$. A vocabulary of 20,000 words was chosen and the remaining tokens were replaced with $\langle$unk$\rangle$.

Following Mitchell and Lapata (2009), we constructed a simple semantic space based on co-occurrence statistics from the BLLIP training set. We used the 2,000 most frequent word types as contexts and a symmetric five word window. Vector components were defined as in Equation (6). We also trained the LDA model on BLLIP, using the Gibb's sampling procedure discussed in Griffiths et al. (2007). We experimented with different numbers of topics on the development set (from 10 to 1,000) and report results on the test set with 100 topics. In our experiments, the hyperparameter α was initialized to .5, and the β word probabilities were initialized randomly.

We integrated our compositional models with a trigram model which we also trained on BLLIP. The model was built using the SRILM toolkit (Stolcke 2002) with backoff and Kneser-Ney smoothing. As our incremental parser we used Roark's (2001) parser trained on sections 2–21 of the Penn Treebank containing 936,017 words. The parser produces prefix probabilities for each word of a sentence which we converted to conditional probabilities by dividing each current probability by the previous one.

Statistical Analysis The statistical analyses in this paper were carried out using linear mixed effects models (LME, Pinheiro and Bates 2000). The latter can be thought of as generalization of linear regression that allows the inclusion of random factors (such as participants or items) as well as fixed factors (e.g., word frequency). In our analyses, we treat participant as a random factor, which means that our models contain an intercept term for each participant, representing the individual differences in the rates at which they read.[3]

We evaluated the effect of adding a factor to a model by comparing the likelihoods of the models with and without that factor. If a χ^2 test on the

[2]The total of all words read by all subjects is 515,020. The pre-processing recommended by Demberg and Keller's (2008) results in a data sets containing 436,000 data points. Removing non-content words leaves 205,922 data points. It only makes sense to consider words that were actually fixated (the eye-tracking measures used are not defined on skipped words), which leaves 162,129 data points. Following Pynte et al. (2008), we require that the previous word was fixated, with 70,051 data points remaining. We exclude words on which the normal left to right movement of gaze had been interrupted, e.g., by blinks and regressions, which results in the final total to 53,704 data points.

[3]Other random factors that are appropriate for our analyses are word and sentence; however, due to the large number of instances for these factors (given that the Dundee corpus contains 51,502 tokens), we were not able to include them: the model fitting algorithm we used (implemented in the R package `lme4`) does not converge for such large models.

Factor	Coefficient
Intercept	$-.011$
Word Length	.264
Launch Distance	.109
Landing Position	.612
Word Frequency	$-.010$
Reading Time of Last Word	.151

Table 1: Coefficients of the baseline LME model for total reading time

Model	Composition	Coefficient
SSS	Additive	$-.03820^{***}$
	Multiplicative	$-.00895^{***}$
LDA	Additive	$-.02500^{***}$
	Multiplicative	$-.00262^{***}$

Table 2: Coefficients of LME models including simple semantic space (SSS) or Latent Dirichlet Allocation (LDA) as factors; $^{***}p < .001$

likelihood ratio is significant, then this indicates that the new factor significantly improves model fit. We also experimented with adding random slopes for participant to the model (in addition to the random intercept); however, this either led to non-convergence of the model fitting procedure, or failed to result in an increase in model fit according to the likelihood ratio test. Therefore, all models reported in the rest of this paper contain random intercept of participants as the sole random factor.

Rather than model raw reading times, we model times on the log scale. This is desirable for a number of reasons. Firstly, the raw reading times tend to have a skew distribution and taking logs produces something closer to normal, which is preferable for modeling. Secondly, the regression equation makes more sense on the log scale as the contribution of each term to raw reading time is multiplicative rather than additive. That is, $log(t) = \sum_i \beta_i x_i$ implies $t = \prod_i e^{\beta_i x_i}$. In particular, the intercept term for each participant now represents a multiplicative factor by which that participant is slower or faster.

5 Results

We computed separate mixed effects models for three dependent variables, namely first fixation duration, first pass duration, and total reading time. We report results for total times throughout, as the results of the other two dependent variables are broadly similar. Our strategy was to first construct a baseline model of low-level factors influencing reading time, and then to take the residuals from that model as the dependent variable in subsequent analyses. In this way we removed the effects of low-level factors before investigating the factors associated with syntactic and semantic constraint. This avoids problems with collinearity between low-level factors and the factors we are interested in (e.g., trigram probability is highly correlated with word frequency). The baseline model contained the factors word length, word frequency, launch distance, landing position, and the reading time for the last fixated word, and its parameter estimates are given in Table 1. To further reduce collinearity, we also centered all fixed factors, both in the baseline model, and in the models fitted on the residuals that we report in the following. Note that some intercorrelations remain between the factors, which we will discuss at the end of Section 5.

Before investigating whether an integrated model of semantic and syntactic constraint improves the goodness of fit over the baseline, we examined the influence of semantic constraint alone. This was necessary as compositional models have not been previously used to model processing difficulty. Besides, replicating Pynte et al.'s (2008) finding, we were also interested in assessing whether the underlying semantic representation (simple semantic space or LDA) and composition function (additive versus multiplicative) modulate reading times differentially.

We built an LME model that predicted the residual reading times of the baseline model using the similarity scores from our composition models as factors. We then carried out a χ^2 test on the likelihood ratio of a model only containing the random factor and the intercept, and a model also containing the semantic factor (cosine similarity). The addition of the semantic factor significantly improves model fit for both the simple semantic space and LDA. This result is observed for both additive and multiplicative composition functions. Our results are summarized in Table 2 which reports the coefficients of the four LME models fitted against the residuals of the baseline model, together with the p-values of the χ^2 test.

Before evaluating our integrated surprisal measure, we evaluated its components *individually* in order to tease their contributions apart. For example, it may be the case that syntactic surprisal is an overwhelmingly better predictor of reading time than semantic surprisal, however we would not be able to detect this by simply adding a factor based on Equation (9) to the baseline model. The

	Factor	SSS Coef	LDA Coef
	$-\log(p)$	.00760***	.00760***
Add	$-\log(\Delta)$	.03810***	.00622***
Add	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	.00953***	.00943***
Mult	$-\log(\Delta)$	.01110***	−.00033
Mult	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	.00882***	.00133

Table 3: Coefficients of nested LME models with the components of SSS or LDA surprisal as factors; only the coefficient of the additional factor at each step are shown

Model	Composition	Coefficient
SSS	Additive	.00804***
	Multiplicative	.00819***
LDA	Additive	.00817***
	Multiplicative	.00640***

Table 4: Coefficients of LME models with integrated surprisal measure (based on SSS or LDA) as factor

integrated surprisal measure can be written as:

$$S = -\log(\lambda p_1 + (1-\lambda)p_2) \quad (10)$$

Where p_2 is the incremental parser probability and p_1 is the product of the semantic component, Δ, and the trigram probability, p. This can be broken down into the sum of two terms:

$$S = -\log(p_1) - \log(\lambda + (1-\lambda)\frac{p_2}{p_1}) \quad (11)$$

Since the first term, $-\log(p_1)$ is itself a product it can also be broken down further:

$$S = -\log(p) - \log(\Delta) - \log(\lambda + (1-\lambda)\frac{p_2}{p_1}) \quad (12)$$

Thus, to evaluate the contribution of the three components to the integrated surprisal measure we fitted nested LME models, i.e., we entered these terms one at a time into a mixed effects model and tested the significance of the improvement in model fit for each additional term.

We again start with an LME model that only contains the random factor and the intercept, with the residuals of the baseline models as the dependent variable. Considering the trigram model first, we find that adding this factor to the model gives a significant improvement in fit. Also adding the semantic component ($-\log(\Delta)$) improves fit further, both for additive and multiplicative composition functions using a simple semantic space. Finally, the addition of the parser probabilities ($\log(\lambda + (1-\lambda)\frac{p_2}{p_1})$) again improves model fit significantly. As far as LDA is concerned, the additive model significantly improves model fit, whereas the multiplicative one does not. These results mirror the findings of Mitchell and Lapata (2009), who report that a multiplicative composition function produced the lowest perplexity for the simple semantic space model, whereas an additive function gave the best perplexity for the LDA space. Table 3 lists the coefficients for the nested models for all four variants of our semantic constraint measure.

Finally, we built a separate LME model where we added the integrated surprisal measure (see Equation (9)) to the model only containing the random factor and the intercept (see Table 4). We did this separately for all four versions of the integrated surprisal measure (SSS, LDA; additive, multiplicative). We find that model fit improved significantly all versions of integrated surprisal.

One technical issue that remains to be discussed is collinearity, i.e., intercorrelations between the factors in a model. The presence of collinearity is problematic, as it can render the model fitting procedure unstable; it can also affect the significance of individual factors. As mentioned in Section 4 we used two techniques to reduce collinearity: residualizing and centering. Table 5 gives an overview of the correlation coefficients for all pairs of factors. It becomes clear that collinearity has mostly been removed; there is a remaining relationship between word length and word frequency, which is expected as shorter words tend to be more frequent. This correlation is not a problem for our analysis, as it is confined to the baseline model. Furthermore, word frequency and trigram probability are highly correlated. Again this is expected, given that the frequencies of unigrams and higher-level n-grams tend to be related. This correlation is taken care of by residualizing, which isolates the two factors: word frequency is part of the baseline model, while trigram probability is part of the separate models that we fit on the residuals. All other correlations are small (with coefficients of .27 or less), with one exception: there is a high correlation between the $-\log(\Delta)$ term and the $\log(\lambda + (1-\lambda)\frac{p_2}{p_1})$ term in the multiplicative LDA model. This collinearity issue may explain the absence of a significant improvement in model fit when these two terms are added to the baseline (see Table 3).

	Factor	Len	Freq	$-\mathrm{l}(p)$	$-\mathrm{l}(\Delta)$
	Frequency	$-.310$			
	$-\log(p)$	.230	$-.700$		
SSS Add	$-\log(\Delta)$	.016	$-.120$	.025	
	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	.024	.036	$-.270$	.065
SSS Mult	$-\log(\Delta)$	$-.015$	$-.110$	.035	
	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	.020	.028	$-.260$	.160
LDA Add	$-\log(\Delta)$	$-.024$	$-.130$	.046	
	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	.005	.014	$-.250$	.030
LDA Mult	$-\log(\Delta)$	$-.120$	.006	$-.046$	
	$\log(\lambda+(1-\lambda)\frac{p_2}{p_1})$	$-.089$	$-.005$	$-.180$	.740

Table 5: Intercorrelations between model factors

6 Discussion

In this paper we investigated the contributions of syntactic and semantic constraint in modeling processing difficulty. Our work departs from previous approaches in that we propose a *single* measure which integrates syntactic and semantic factors. Evaluation on an eye-tracking corpus shows that our measure predicts reading time better than a baseline model that captures low-level factors in reading (word length, landing position, etc.). Crucially, we were able to show that the semantic component of our measure improves reading time predictions over and above a model that includes syntactic measures (based on a trigram model and incremental parser). This means that semantic costs are a significant predictor of reading time *in addition* to the well-known syntactic surprisal.

An open issue is whether a single, integrated measure (as evaluated in Table 4) fits the eye-movement data significantly better than separate measures for trigram, syntactic, and semantic surprisal (as evaluated in Table 3. However, we are not able to investigate this hypothesis: our approach to testing the significance of factors requires nested models; the log-likelihood test (see Section 4) is only able to establish whether adding a factor to a model improves its fit; it cannot compare models with disjunct sets of factors (such as a model containing the integrated surprisal measure and one containing the three separate ones). However, we would argue that a single, integrated measure that captures human predictive processing is preferable over a collection of separate measures. It is conceptually simpler (as it is more parsimonious), and is also easier to use in applications (such as readability prediction). Finally, an integrated measure requires less parameters; our definition of surprisal in 12 is simply the sum of the trigram, syntactic, and semantic components. An LME model containing separate factors, on the other hand, requires a coefficient for each of them, and thus has more parameters.

In evaluating our model, we adopted a broad coverage approach using the reading time data from a naturalistic corpus rather than artificially constructed experimental materials. In doing so, we were able to compare different syntactic and semantic costs on the same footing. Previous analyses of semantic constraint have been conducted on different eye-tracking corpora (Dundee and Embra Corpus) and on different languages (English, French). Moreover, comparisons of the individual contributions of syntactic and semantic factors were generally absent from the literature. Our analysis showed that both of these factors can be captured by our integrated surprisal measure which is uniformly probabilistic and thus preferable to modeling semantic and syntactic costs disjointly using a mixture of probabilistic and non-probabilistic measures.

An interesting question is which aspects of semantics our model is able to capture, i.e., why does the combination of LSA or LDA representations with an incremental parser yield a better fit of the behavioral data. In the psycholinguistic literature, various types of semantic information have been investigated: lexical semantics (word senses, selectional restrictions, thematic roles), sentential semantics (scope, binding), and discourse semantics (coreference and coherence); see Keller (2010) of a detailed discussion. We conjecture that our model is mainly capturing lexical semantics (through the vector space representation of words) and sentential semantics (through the multiplication or addition of words). However, discourse coreference effects (such as the ones reported by Altmann and Steedman (1988) and much subsequent work) are probably not amenable to a treatment in terms of vector space semantics; an explicit representation of discourse entities and coreference relations is required (see Dubey 2010 for a model of human sentence processing that can handle coreference).

A key objective for future work will be to investigate models that integrate semantic constraint with syntactic predictions more tightly. For example, we could envisage a parser that uses semantic representations to guide its search, e.g., by pruning syntactic analyses that have a low semantic probability. At the same time, the semantic model should have access to syntactic information, i.e., the composition of word representations should take their syntactic relationships into account, rather than just linear order.

References

ACL. 2010. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala.

Altmann, Gerry T. M. and Yuki Kamide. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition* 73:247–264.

Altmann, Gerry T. M. and Mark J. Steedman. 1988. Interaction with context during human sentence processing. *Cognition* 30(3):191–238.

Bellegarda, Jerome R. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE* 88(8):1279–1296.

Berry, Michael W., Susan T. Dumais, and Gavin W. O'Brien. 1995. Using linear algebra for intelligent information retrieval. *SIAM review* 37(4):573–595.

Bever, Thomas G. 1970. The cognitive basis for linguistic strutures. In J. R. Hayes, editor, *Cognition and the Development of Language*, Wiley, New York, pages 279–362.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Bullinaria, John A. and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods* 39:510–526.

Clifton, Charles, Adrian Staub, and Keith Rayner. 2007. Eye movement in reading words and sentences. In R V Gompel, M Fisher, W Murray, and R L Hill, editors, *Eye Movements: A Window in Mind and Brain*, Elsevier, pages 341–372.

Coccaro, Noah and Daniel Jurafsky. 1998. Towards better integration of semantic predictors in satistical language modeling. In *Proceedings of the 5th International Conference on Spoken Language Processing*. Sydney, Australia, pages 2403–2406.

Demberg, Vera and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition* 101(2):193–210.

Dubey, Amit. 2010. The influence of discourse on syntax: A psycholinguistic model of sentence processing. In ACL.

Ferrara Boston, Marisa, John Hale, Reinhold Kliegl, Umesh Patil, and Shravan Vasishth. 2008. Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam Sentence Corpus. *Journal of Eye Movement Research* 2(1):1–12.

Frank, Stefan L. 2009. Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*. Austin, TX, pages 139–1144.

Gibson, Edward. 2000. Dependency locality theory: A distance-dased theory of linguistic complexity. In Alec Marantz, Yasushi Miyashita, and Wayne O'Neil, editors, *Image, Language, Brain: Papers from the First Mind Articulation Project Symposium*, MIT Press, Cambridge, MA, pages 95–126.

Gildea, Daniel and Thomas Hofmann. 1999. Topic-based language models using EM. In *Proceedings of the 6th European Conference on Speech Communiation and Technology*. Budapest, Hungary, pages 2167–2170.

Griffiths, Thomas L., Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review* 114(2):211–244.

Hale, John. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd Conference of the North American Chapter of the Association*. Association for Computational Linguistics, Pittsburgh, PA, volume 2, pages 159–166.

Keller, Frank. 2010. Cognitively plausible models of human language processing. In ACL.

Kennedy, Alan and Joel Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research* 45:153–168.

Konieczny, Lars. 2000. Locality and parsing complexity. *Journal of Psycholinguistic Research* 29(6):627–645.

Landauer, Thomas K. and Susan T. Dumais. 1997. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review* 104(2):211–240.

Levy, Roger. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3):1126–1177.

Marslen-Wilson, William D. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature* 244:522–523.

McDonald, Scott. 2000. *Environmental Determinants of Lexical Processing Effort*. Ph.D. thesis, University of Edinburgh.

McDonald, Scott and Chris Brew. 2004. A distributional model of semantic context effects in lexical processing. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics*. Barcelona, Spain, pages 17–24.

McDonald, Scott A. and Richard C. Shillcock. 2003. Low-level predictive inference in reading: The influence of transitional probabilities on eye movements. *Vision Research* 43:1735–1751.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*. Columbus, OH, pages 236–244.

Mitchell, Jeff and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore, pages 430–439.

Narayanan, Srini and Daniel Jurafsky. 2002. A Bayesian model predicts human parse preference and reading time in sentence processing. In Thomas G. Dietterich, Sue Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, pages 59–65.

Padó, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics* 33(2):161–199.

Padó, Ulrike, Matthew W. Crocker, and Frank Keller. 2009. A probabilistic model of semantic plausibility in sentence processing. *Cognitive Science* 33(5):794–838.

Pinheiro, Jose C. and Douglas M. Bates. 2000. *Mixed-effects Models in S and S-PLUS*. Springer, New York.

Pinker, Steven. 1994. *The Language Instinct: How the Mind Creates Language*. HarperCollins, New York.

Plate, Tony A. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks* 6(3):623–641.

Pynte, Joel, Boris New, and Alan Kennedy. 2008. On-line contextual influences during reading normal text: A multiple-regression analysis. *Vision Research* 48:2172–2183.

Rayner, Keith. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 124(3):372–422.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics* 27(2):249–276.

Roark, Brian, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 324–333.

Smolensky, Paul. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46:159–216.

Stanovich, Kieth E. and Richard F. West. 1981. The effect of sentence context on ongoing word recognition: Tests of a two-pricess theory. *Journal of Experimental Psychology: Human Perception and Performance* 7:658–672.

Staub, Adrian and Charles Clifton. 2006. Syntactic prediction in language comprehension: Evidence from either . . .or. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 32:425–436.

Steyvers, Mark and Tom Griffiths. 2007. Probabilistic topic models. In T. Landauer, D. McNamara, S Dennis, and W Kintsch, editors, *A Handbook of Latent Semantic Analysis*, Psychology Press.

Stolcke, Andreas. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of the Internatinal Conference on Spoken Language Processing*. Denver, Colorado.

Sturt, Patrick and Vincenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science* 29(2):291–305.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268:1632–1634.

van Berkum, Jos J. A., Colin M. Brown, and Peter Hagoort. 1999. Early referential context effects in sentence processing: Evidence from event-related brain potentials. *Journal of Memory and Language* 41:147–182.

Wright, Barton and Merrill F. Garrett. 1984. Lexical decision in sentences: Effects of syntactic structure. *Memory and Cognition* 12:31–45.

Rebanking CCGbank for improved NP interpretation

Matthew Honnibal and James R. Curran
School of Information Technologies
University of Sydney
NSW 2006, Australia
{mhonn,james}@it.usyd.edu.au

Johan Bos
University of Groningen
The Netherlands
bos@meaningfactory.com

Abstract

Once released, treebanks tend to remain unchanged despite any shortcomings in their depth of linguistic analysis or coverage of specific phenomena. Instead, separate resources are created to address such problems. In this paper we show how to improve the quality of a treebank, by integrating resources and implementing improved analyses for specific constructions.

We demonstrate this *rebanking* process by creating an updated version of CCGbank that includes the predicate-argument structure of both verbs and nouns, base-NP brackets, verb-particle constructions, and restrictive and non-restrictive nominal modifiers; and evaluate the impact of these changes on a statistical parser.

1 Introduction

Progress in natural language processing relies on direct comparison on shared data, discouraging improvements to the evaluation data. This means that we often spend years competing to reproduce partially incorrect annotations. It also encourages us to approach related problems as discrete tasks, when a new data set that adds deeper information establishes a new incompatible evaluation.

Direct comparison has been central to progress in statistical parsing, but it has also caused problems. Treebanking is a difficult engineering task: coverage, cost, consistency and granularity are all competing concerns that must be balanced against each other when the annotation scheme is developed. The difficulty of the task means that we ought to view treebanking as an ongoing process akin to grammar development, such as the many years of work on the ERG (Flickinger, 2000).

This paper demonstrates how a treebank can be *rebanked* to incorporate novel analyses and information from existing resources. We chose to work on CCGbank (Hockenmaier and Steedman, 2007), a Combinatory Categorial Grammar (Steedman, 2000) treebank acquired from the Penn Treebank (Marcus et al., 1993). This work is equally applicable to the corpora described by Miyao et al. (2004), Shen et al. (2008) or Cahill et al. (2008).

Our first changes integrate four previously suggested improvements to CCGbank. We then describe a novel CCG analysis of NP predicate-argument structure, which we implement using NomBank (Meyers et al., 2004). Our analysis allows the distinction between core and peripheral arguments to be represented for predicate nouns.

With this distinction, an entailment recognition system could recognise that *Google's acquisition of YouTube* entailed *Google acquired YouTube*, because equivalent predicate-argument structures are built for both. Our analysis also recovers non-local dependencies mediated by nominal predicates; for instance, *Google* is the agent of *acquire* in *Google's decision to acquire YouTube*.

The rebanked corpus extends CCGbank with:

1. NP brackets from Vadas and Curran (2008);
2. Restored and normalised punctuation;
3. Propbank-derived verb subcategorisation;
4. Verb particle structure drawn from Propbank;
5. Restrictive and non-restrictive adnominals;
6. Reanalyses to promote better head-finding;
7. Nombank-derived noun subcategorisation.

Together, these changes modify 30% of the labelled dependencies in CCGbank, demonstrating how multiple resources can be brought together in a single, richly annotated corpus. We then train and evaluate a parser for these changes, to investigate their impact on the accuracy of a state-of-the-art statistical CCG parser.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 207–215,
Uppsala, Sweden, 11-16 July 2010.

2 Background and motivation

Formalisms like HPSG (Pollard and Sag, 1994), LFG (Kaplan and Bresnan, 1982), and CCG (Steedman, 2000) are *linguistically motivated* in the sense that they attempt to explain and predict the limited variation found in the grammars of natural languages. They also attempt to specify how grammars construct semantic representations from surface strings, which is why they are sometimes referred to as *deep* grammars. Analyses produced by these formalisms can be more detailed than those produced by skeletal phrase-structure parsers, because they produce fully specified predicate-argument structures.

Unfortunately, statistical parsers do not take advantage of this potential detail. Statistical parsers induce their grammars from corpora, and the corpora for linguistically motivated formalisms currently do not contain high quality predicate-argument annotation, because they were derived from the Penn Treebank (PTB Marcus et al., 1993). Manually written grammars for these formalisms, such as the ERG HPSG grammar (Flickinger, 2000) and the XLE LFG grammar (Butt et al., 2006) produce far more detailed and linguistically correct analyses than any English statistical parser, due to the comparatively coarse-grained annotation schemes of the corpora statistical parsers are trained on. While rule-based parsers use grammars that are carefully engineered (e.g. Oepen et al., 2004), and can be updated to reflect the best linguistic analyses, statistical parsers have so far had to take what they are given.

What we suggest in this paper is that a treebank's grammar need not last its lifetime. For a start, there have been many annotations of the PTB that add much of the extra information needed to produce very high quality analyses for a linguistically motivated grammar. There are also other transformations which can be made with no additional information. That is, sometimes the existing trees allow transformation rules to be written that improve the quality of the grammar.

Linguistic theories are constantly changing, which means that there is a substantial lag between what we (think we) understand of grammar and the annotations in our corpora. The grammar engineering process we describe, which we dub *rebanking*, is intended to reduce this gap, tightening the feedback loop between formal and computational linguistics.

2.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG; Steedman, 2000) is a lexicalised grammar, which means that all grammatical dependencies are specified in the lexical entries and that the production of derivations is governed by a small set of rules.

Lexical categories are either atomic (S, NP, PP, N), or a functor consisting of a result, directional slash, and argument. For instance, *in* might head a PP-typed constituent with one NP-typed argument, written as PP/NP.

A category can have a functor as its result, so that a word can have a complex valency structure. For instance, a verb phrase is represented by the category $S\backslash NP$: it is a function from a leftward NP (a subject) to a sentence. A transitive verb requires an object to become a verb phrase, producing the category $(S\backslash NP)/NP$.

A CCG grammar consists of a small number of schematic rules, called combinators. CCG extends the basic application rules of pure categorial grammar with (generalised) composition rules and type raising. The most common rules are:

$$\begin{array}{rrcll} X/Y & Y & \Rightarrow & X & (>) \\ Y & X\backslash Y & \Rightarrow & X & (<) \\ X/Y & Y/Z & \Rightarrow & X/Z & (>\mathbf{B}) \\ Y\backslash Z & X\backslash Y & \Rightarrow & X\backslash Z & (<\mathbf{B}) \\ Y/Z & X\backslash Y & \Rightarrow & X/Z & (<\mathbf{B}_{\times}) \end{array}$$

CCGbank (Hockenmaier and Steedman, 2007) extends this compact set of combinatory rules with a set of type-changing rules, designed to strike a better balance between sparsity in the category set and ambiguity in the grammar. We mark type-changing rules **TC** in our derivations.

In wide-coverage descriptions, categories are generally modelled as typed-feature structures (Shieber, 1986), rather than atomic symbols. This allows the grammar to include a notion of headedness, and to unify under-specified features.

We occasionally must refer to these additional details, for which we employ the following notation. Features are annotated in square-brackets, e.g. $S[dcl]$. Head-finding indices are annotated on categories in subscripts, e.g. $(NP_y\backslash NP_y)/NP_z$. The index of the word the category is assigned to is left implicit. We will sometimes also annotate derivations with the heads of categories as they are being built, to help the reader keep track of what lexemes have been bound to which categories.

3 Combining CCGbank corrections

There have been a few papers describing corrections to CCGbank. We bring these corrections together for the first time, before building on them with our further changes.

3.1 Compound noun brackets

Compound noun phrases can nest inside each other, creating bracketing ambiguities:

(1) (crude oil) prices

(2) crude (oil prices)

The structure of such compound noun phrases is left underspecified in the Penn Treebank (PTB), because the annotation procedure involved stitching together partial parses produced by the Fidditch parser (Hindle, 1983), which produced flat brackets for these constructions. The bracketing decision was also a source of annotator disagreement (Bies et al., 1995).

When Hockenmaier and Steedman (2002) went to acquire a CCG treebank from the PTB, this posed a problem. There is no equivalent way to leave these structures under-specified in CCG, because derivations must be binary branching. They therefore employed a simple heuristic: assume all such structures branch to the right. Under this analysis, *crude oil* is not a constituent, producing an incorrect analysis as in (1).

Vadas and Curran (2007) addressed this by manually annotating all of the ambiguous noun phrases in the PTB, and went on to use this information to correct 20,409 dependencies (1.95%) in CCGbank (Vadas and Curran, 2008). Our changes build on this corrected corpus.

3.2 Punctuation corrections

The syntactic analysis of punctuation is notoriously difficult, and punctuation is not always treated consistently in the Penn Treebank (Bies et al., 1995). Hockenmaier (2003) determined that quotation marks were particularly problematic, and therefore removed them from CCGbank altogether. We use the process described by Tse and Curran (2008) to restore the quotation marks and shift commas so that they always attach to the constituent to their left. This allows a grammar rule to be removed, preventing a great deal of spurious ambiguity and improving the speed of the C&C parser (Clark and Curran, 2007) by 37%.

3.3 Verb predicate-argument corrections

Semantic role descriptions generally recognise a distinction between core arguments, whose role comes from a set specific to the predicate, and peripheral arguments, who have a role drawn from a small, generic set. This distinction is represented in the surface syntax in CCG, because the category of a verb must specify its argument structure. In (3) *as a director* is annotated as a complement; in (4) it is an adjunct:

(3) *He* *joined* *as a director*
NP $(S\backslash NP)/PP$ PP

(4) *He* *joined* *as a director*
NP $S\backslash NP$ $(S\backslash NP)\backslash(S\backslash NP)$

CCGbank contains noisy complement and adjunct distinctions, because they were drawn from PTB function labels which imperfectly represent the distinction. In our previous work we used Propbank (Palmer et al., 2005) to convert 1,543 complements to adjuncts and 13,256 adjuncts to complements (Honnibal and Curran, 2007). If a constituent such as *as a director* received an adjunct category, but was labelled as a core argument in Propbank, we changed it to a complement, using its head's part-of-speech tag to infer its constituent type. We performed the equivalent transformation to ensure all peripheral arguments of verbs were analysed as adjuncts.

3.4 Verb-particle constructions

Propbank also offers reliable annotation of verb-particle constructions. This was not available in the PTB, so Hockenmaier and Steedman (2007) annotated all intransitive prepositions as adjuncts:

(5) *He* *woke* *up*
NP $S\backslash NP$ $(S\backslash NP)\backslash(S\backslash NP)$

We follow Constable and Curran (2009) in exploiting the Propbank annotations to add verb-particle distinctions to CCGbank, by introducing a new atomic category PT for particles, and changing their status from adjuncts to complements:

(6) *He* *woke* *up*
NP $(S\backslash NP)/PT$ PT

This analysis could be improved by adding extra head-finding logic to the verbal category, to recognise the multi-word expression as the head.

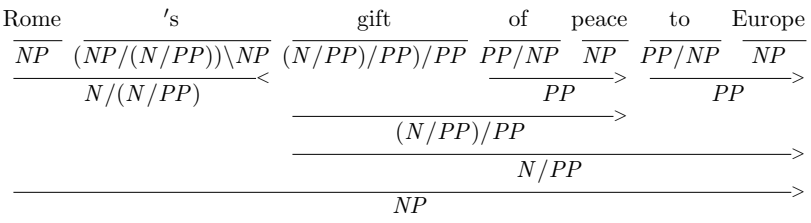


Figure 1: Deverbal noun predicate with agent, patient and beneficiary arguments.

4 Noun predicate-argument structure

Many common nouns in English can receive optional complements and adjuncts, realised by prepositional phrases, genitive determiners, compound nouns, relative clauses, and for some nouns, complementised clauses. For example, deverbal nouns generally have argument structures similar to the verbs they are derived from:

(7) Rome's destruction of Carthage

(8) Rome destroyed Carthage

The semantic roles of *Rome* and *Carthage* are the same in (7) and (8), but the noun cannot case-mark them directly, so *of* and the genitive clitic are pressed into service. The semantic role depends on both the predicate and subcategorisation frame:

(9) Carthage's$_{\mathrm{p}}$ destruction$_{\mathrm{Pred.}}$

(10) Rome's$_{\mathrm{a}}$ destruction$_{\mathrm{Pred.}}$ of Carthage$_{\mathrm{p}}$

(11) Rome's$_{\mathrm{a}}$ gift$_{\mathrm{Pred.}}$

(12) Rome's$_{\mathrm{a}}$ gift$_{\mathrm{Pred.}}$ of peace$_{\mathrm{p}}$ to Europe$_{\mathrm{b}}$

In (9), the genitive introduces the patient, but when the patient is supplied by the PP, it instead introduces the agent. The mapping differs for *gift*, where the genitive introduces the agent.

Peripheral arguments, which supply generically available modifiers of time, place, cause, quality etc, can be realised by pre- and post-modifiers:

(13) The portrait *in the Louvre*

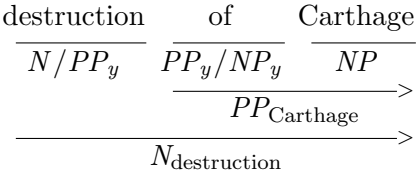
(14) The *fine* portrait

(15) *The Louvre's* portraits

These are distinct from core arguments because their interpretation does not depend on the predicate. The ambiguity can be seen in an NP such as *The nobleman's portrait*, where the genitive could mark possession (peripheral), or it could introduce the patient (core). The distinction between core and peripheral arguments is particularly difficult for compound nouns, as pre-modification is very productive in English.

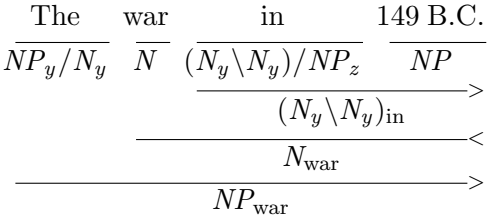
4.1 CCG analysis

We designed our analysis for transparency between the syntax and the predicate-argument structure, by stipulating that all and only the core arguments should be syntactic arguments of the predicate's category. This is fairly straightforward for arguments introduced by prepositions:

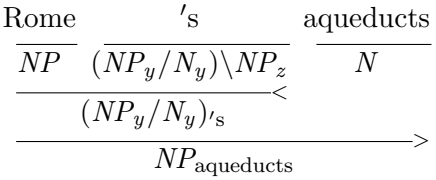


In our analysis, the head of *of Carthage* is *Carthage*, as *of* is assumed to be a semantically transparent case-marker. We apply this analysis to prepositional phrases that provide arguments to verbs as well — a departure from CCGbank.

Prepositional phrases that introduce peripheral arguments are analysed as syntactic adjuncts:



Adjunct prepositional phrases remain headed by the preposition, as it is the preposition's semantics that determines whether they function as temporal, causal, spatial etc. arguments. We follow Hockenmaier and Steedman (2007) in our analysis of genitives which realise peripheral arguments, such as the literal possessive:



Arguments introduced by possessives are a little trickier, because the genitive also functions as a determiner. We achieve this by having the noun subcategorise for the argument, which we type PP, and having the possessive subcategorise for the unsaturated noun to ultimately produce an NP:

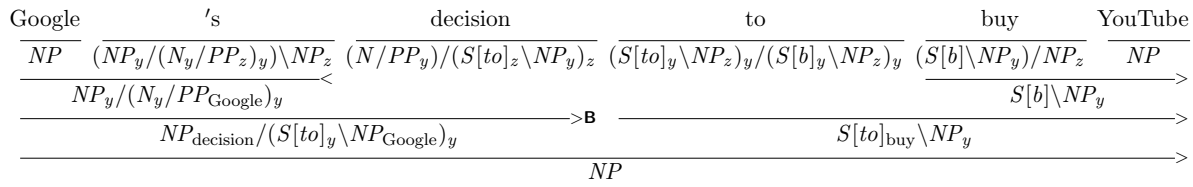
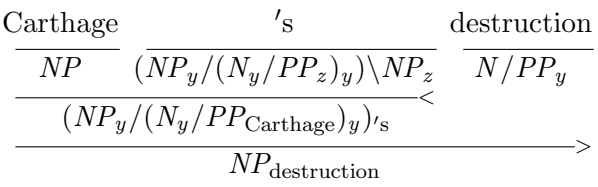


Figure 2: The coindexing on *decision*'s category allows the hard-to-reach agent of *buy* to be recovered. A non-normal form derivation is shown so that instantiated variables can be seen.



In this analysis, we regard the genitive clitic as a case-marker that performs a movement operation roughly analogous to WH-extraction. Its category is therefore similar to the one used in object extraction, $(N\backslash N)/(S/NP)$. Figure 1 shows an example with multiple core arguments.

This analysis allows recovery of verbal arguments of nominalised raising and control verbs, a construction which both Gildea and Hockenmaier (2003) and Boxwell and White (2008) identify as a problem case when aligning Propbank and CCGbank. Our analysis accommodates this construction effortlessly, as shown in Figure 2. The category assigned to *decision* can coindex the missing NP argument of *buy* with its own PP argument. When that argument is supplied by the genitive, it is also supplied to the verb, *buy*, filling its dependency with its agent, *Google*. This argument would be quite difficult to recover using a shallow syntactic analysis, as the path would be quite long. There are 494 such verb arguments mediated by nominal predicates in Sections 02-21.

These analyses allow us to draw complement/adjunct distinctions for nominal predicates, so that the surface syntax takes us very close to a full predicate-argument analysis. The only information we are not specifying in the syntactic analysis are the role labels assigned to each of the syntactic arguments. We could go further and express these labels in the syntax, producing categories like $(N/PP\{0\}_y)/PP\{1\}_z$ and $(N/PP\{1\}_y)/PP\{0\}_z$, but we expect that this would cause sparse data problems given the limited size of the corpus. This experiment would be an interesting subject of future work.

The only local core arguments that we do not annotate as syntactic complements are compound nouns, such as *decision makers*. We avoided these arguments because of the productivity of noun-noun compounding in English, which makes these argument structures very difficult to recover.

We currently do not have an analysis that allows support verbs to supply noun arguments, so we do not recover any of the long-range dependency structures described by Meyers et al. (2004).

4.2 Implementation and statistics

Our analysis requires semantic role labels for each argument of the nominal predicates in the Penn Treebank — precisely what NomBank (Meyers et al., 2004) provides. We can therefore draw our distinctions using the process described in our previous work, Honnibal and Curran (2007).

NomBank follows the same format as Propbank, so the procedure is exactly the same. First, we align CCGbank and the Penn Treebank, and produce a version of NomBank that refers to CCGbank nodes. We then assume that any prepositional phrase or genitive determiner annotated as a core argument in NomBank should be analysed as a complement, while peripheral arguments and adnominals that receive no semantic role label at all are analysed as adjuncts.

We converted 34,345 adnominal prepositional phrases to complements, leaving 18,919 as adjuncts. The most common preposition converted was *of*, which was labelled as a core argument 99.1% of the 19,283 times it occurred as an adnominal. The most common adjunct preposition was *in*, which realised a peripheral argument in 59.1% of its 7,725 occurrences.

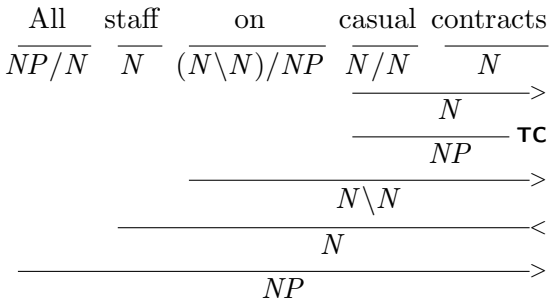
The frequent prepositions were more skewed towards core arguments. 73% of the occurrences of the 5 most frequent prepositions (*of*, *in*, *for*, *on* and *to*) realised peripheral arguments, compared with 53% for other prepositions.

Core arguments were also more common than peripheral arguments for possessives. There are 20,250 possessives in the corpus, of which 75% were converted to complements. The percentage was similar for both personal pronouns (such as *his*) and genitive phrases (such as *the boy's*).

5 Adding restrictivity distinctions

Adnominals can have either a restrictive or a non-restrictive (appositional) interpretation, determining the potential reference of the noun phrase it modifies. This ambiguity manifests itself in whether prepositional phrases, relative clauses and other adnominals are analysed as modifiers of either N or NP, yielding a restrictive or non-restrictive interpretation respectively.

In CCGbank, all adnominals attach to NPs, producing non-restrictive interpretations. We therefore move restrictive adnominals to N nodes:



This corrects the previous interpretation, which stated that there were no permanent staff.

5.1 Implementation and statistics

The Wall Street Journal's style guide mandates that this attachment ambiguity be managed by bracketing non-restrictive relatives with commas (Martin, 2002, p. 82), as in *casual staff, who have no health insurance, support it*. We thus use punctuation to make the attachment decision.

All $NP\backslash NP$ modifiers that are not preceded by punctuation were moved to the lowest N node possible and relabelled $N\backslash N$. We select the lowest (i.e. closest to leaf) N node because some adjectives, such as *present* or *former*, require scope over the qualified noun, making it safer to attach the adnominal first.

Some adnominals in CCGbank are created by the $S\backslash NP \rightarrow NP\backslash NP$ unary type-changing rule, which transforms reduced relative clauses. We introduce a $S\backslash NP \rightarrow N\backslash N$ in its place, and add a binary rule cued by punctuation to handle the relatively rare non-restrictive reduced relative clauses.

The rebanked corpus contains 34,134 $N\backslash N$ restrictive modifiers, and 9,784 non-restrictive modifiers. Most (61%) of the non-restrictive modifiers were relative clauses.

6 Reanalysing partitive constructions

True partitive constructions consist of a quantifier (16), a cardinal (17) or demonstrative (18) applied to an NP via *of*. There are similar constructions headed by common nouns, as in (19):

(16) Some of us

(17) Four of our members

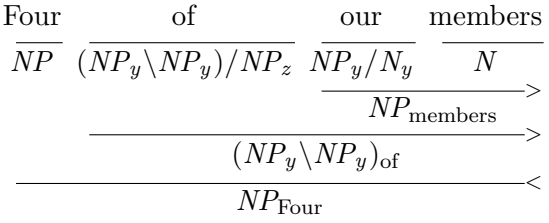
(18) Those of us who smoke

(19) A glass of wine

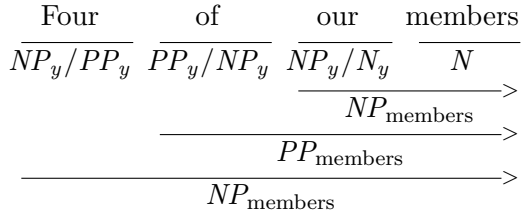
We regard the common noun partitives as headed by the initial noun, such as *glass*, because this noun usually controls the number agreement. We therefore analyse these cases as nouns with prepositional arguments. In (19), *glass* would be assigned the category N/PP.

True partitive constructions are different, however: they are always headed by the head of the NP supplied by *of*. The construction is quite common, because it provides a way to quantify or apply two different determiners.

Partitive constructions are not given special treatment in the PTB, and were analysed as noun phrases with a PP modifier in CCGbank:



This analysis does not yield the correct semantics, and may even hurt parser performance, because the head of the phrase is incorrectly assigned. We correct this with the following analysis, which takes the head from the NP argument of the PP:



The cardinal is given the category NP/PP, in analogy with the standard determiner category which is a function from a noun to a noun phrase (NP/N).

Corpus	L. DEPS	U. DEPS	CATS
+NP brackets	97.2	97.7	98.5
+Quotes	97.2	97.7	98.5
+Propbank	93.0	94.9	96.7
+Particles	92.5	94.8	96.2
+Restrictivity	79.5	94.4	90.6
+Part. Gen.	76.1	90.1	90.4
+NP Pred-Arg	70.6	83.3	84.8

Table 1: Effect of the changes on CCGbank, by percentage of dependencies and categories left unchanged in Section 00.

Corpus	CATS	Cats ≥ 10	CATS/WORD
CCGbank	1286	425	8.6
+NP brackets	1298	429	8.9
+Quotes	1300	431	8.8
+Propbank	1342	433	8.9
+Particles	1405	458	9.1
+Restrictivity	1447	471	9.3
+Part. Gen.	1455	474	9.5
+NP Pred-Arg	1574	511	10.1

Table 2: Effect of the changes on the size of the lexicon.

6.1 Implementation and Statistics

We detect this construction by identifying NPs post-modified by an *of* PP. The NP's head must either have the POS tag CD, or be one of the following words, determined through manual inspection of Sections 02-21:

> *all, another, average, both, each, another, any, anything, both, certain, each, either, enough, few, little, most, much, neither, nothing, other, part, plenty, several, some, something, that, those.*

Having identified the construction, we simply relabel the NP to NP/PP, and the $NP\backslash NP$ adnominal to PP. We identified and reanalysed 3,010 partitive genitives in CCGbank.

7 Similarity to CCGbank

Table 1 shows the percentage of labelled dependencies (L. Deps), unlabelled dependencies (U. Deps) and lexical categories (Cats) that remained the same after each set of changes.

A labelled dependency is a 4-tuple consisting of the head, the argument, the lexical category of the head, and the argument slot that the dependency fills. For instance, the subject fills slot 1 and the object fills slot 2 on the transitive verb category $(S\backslash NP)/NP$. There are more changes to labelled dependencies than lexical categories because one lexical category change alters all of the dependencies headed by a predicate, as they all depend on its lexical category. Unlabelled dependencies consist of only the head and argument.

The biggest changes were those described in Sections 4 and 5. After the addition of nominal predicate-argument structure, over 50% of the labelled dependencies were changed. Many of these changes involved changing an adjunct to a complement, which affects the unlabelled dependencies because the head and argument are inverted.

8 Lexicon statistics

Our changes make the grammar sensitive to new distinctions, which increases the number of lexical categories required. Table 2 shows the number of lexical categories (Cats), the number of lexical categories that occur at least 10 times in Sections 02-21 (Cats ≥ 10), and the average number of categories available for assignment to each token in Section 00 (Cats/Word). We followed Clark and Curran's (2007) process to determine the set of categories a word could receive, which includes a part-of-speech back-off for infrequent words.

The lexicon steadily grew with each set of changes, because each added information to the corpus. The addition of quotes only added two categories (LQU and RQU), and the addition of the quote tokens slightly decreased the average categories per word. The Propbank and verb-particle changes both introduced rare categories for complicated, infrequent argument structures.

The NP predicate-argument structure modifications added the most information. Head nouns were previously guaranteed the category N in CCGbank; possessive clitics always received the category $(NP/N)\backslash NP$; and possessive personal pronouns were always NP/N. Our changes introduce new categories for these frequent tokens, which meant a substantial increase in the number of possible categories per word.

9 Parsing Evaluation

Some of the changes we have made correct problems that have caused the performance of a statistical CCG parser to be over-estimated. Other changes introduce new distinctions, which a parser may or may not find difficult to reproduce. To investigate these issues, we trained and evaluated the C&C CCG parser on our rebanked corpora.

The experiments were set up as follows. We used the highest scoring configuration described by Clark and Curran (2007), the hybrid dependency model, using gold-standard POS tags. We followed Clark and Curran in excluding sentences that could not be parsed from the evaluation. All models obtained similar coverage, between 99.0 and 99.3%. The parser was evaluated using depen-

Corpus	WSJ 00			WSJ 23		
	LF	UF	CAT	LF	UF	CAT
CCGbank	87.2	92.9	94.1	87.7	93.0	94.4
+NP brackets	86.9	92.8	93.8	87.3	92.8	93.9
+Quotes	86.8	92.7	93.9	87.1	92.6	94.0
+Propbank	86.7	92.6	94.0	87.0	92.6	94.0
+Particles	86.4	92.5	93.8	86.8	92.6	93.8
All Rebanking	84.2	91.2	91.9	84.7	91.3	92.2

Table 3: Parser evaluation on the rebanked corpora.

Corpus	Rebanked		CCGbank	
	LF	UF	LF	UF
+NP brackets	86.45	92.36	86.52	92.35
+Quotes	86.57	92.40	86.52	92.35
+Propbank	87.76	92.96	87.74	92.99
+Particles	87.50	92.77	87.67	92.93
All Rebanking	87.23	92.71	88.02	93.51

Table 4: Comparison of parsers trained on CCGbank and the rebanked corpora, using dependencies that occur in both.

dencies generated from the gold-standard derivations (Boxwell, p.c., 2010).

Table 3 shows the accuracy of the parser on Sections 00 and 23. The parser scored slightly lower as the NP brackets, Quotes, Propbank and Particles corrections were added. This apparent decline in performance is at least partially an artefact of the evaluation. CCGbank contains some dependencies that are trivial to recover, because Hockenmaier and Steedman (2007) was forced to adopt a strictly right-branching analysis for NP brackets.

There was a larger drop in accuracy on the fully rebanked corpus, which included our analyses of restrictivity, partitive constructions and noun predicate-argument structure. This might also be explained by the evaluation, as the rebanked corpus includes much more fine-grained distinctions. The labelled dependencies evaluation is particularly sensitive to this, as a single category change affects multiple dependencies. This can be seen in the smaller gap in category accuracy.

We investigated whether the differences in performance were due to the different evaluation data by comparing the parsers' performance against the original parser on the dependencies they agreed upon, to allow direct comparison. To do this, we extracted the CCGbank intersection of each corpus's Section 00 dependencies.

Table 4 compares the labelled and unlabelled recall of the rebanked parsers we trained against the CCGbank parser on these intersections. Note that each row refers to a different intersection, so results are not comparable between rows. This comparison shows that the declines in accuracy seen in Table 3 were largely confined to the corrected dependencies. The parser's performance remained fairly stable on the dependencies left unchanged.

The rebanked parser performed 0.8% worse than the CCGbank parser on the intersection dependencies, suggesting that the fine-grained distinctions we introduced did cause some sparse data problems. However, we did not change any of the parser's maximum entropy features or hyperparameters, which are tuned for CCGbank.

10 Conclusion

Research in natural language understanding is driven by the datasets that we have available. The most cited computational linguistics work to date is the Penn Treebank (Marcus et al., 1993)[1]. Propbank (Palmer et al., 2005) has also been very influential since its release, and NomBank has been used for semantic dependency parsing in the CoNLL 2008 and 2009 shared tasks.

This paper has described how these resources can be jointly exploited using a linguistically motivated theory of syntax and semantics. The semantic annotations provided by Propbank and NomBank allowed us to build a corpus that takes much greater advantage of the semantic transparency of a deep grammar, using careful analyses and phenomenon-specific conversion rules.

The major areas of CCGbank's grammar left to be improved are the analysis of comparatives, and the analysis of named entities. English comparatives are diverse and difficult to analyse. Even the XTAG grammar (Doran et al., 1994), which deals with the major constructions of English in enviable detail, does not offer a full analysis of these phenomena. Named entities are also difficult to analyse, as many entity types obey their own specific grammars. This is another example of a phenomenon that could be analysed much better in CCGbank using an existing resource, the BBN named entity corpus.

Our rebanking has substantially improved CCGbank, by increasing the granularity and linguistic fidelity of its analyses. We achieved this by exploiting existing resources and crafting novel analyses. The process we have demonstrated can be used to train a parser that returns dependencies that abstract away as much surface syntactic variation as possible — including, now, even whether the predicate and arguments are expressed in a noun phrase or a full clause.

[1] http://clair.si.umich.edu/clair/anthology/rankings.cgi

Acknowledgments

James Curran was supported by Australian Research Council Discovery grant DP1097291 and the Capital Markets Cooperative Research Centre.

The parsing evaluation for this paper would have been much more difficult without the assistance of Stephen Boxwell, who helped generate the gold-standard dependencies with his software.

We are also grateful to the members of the CCG technicians mailing list for their help crafting the analyses, particularly Michael White, Mark Steedman and Dennis Mehay.

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, MS-CIS-95-06, University of Pennsylvania, Philadelphia, PA, USA.

Stephen Boxwell and Michael White. 2008. Projecting propbank roles onto the CCGbank. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 3112–3117. European Language Resources Association (ELRA), Marrakech, Morocco.

Miriam Butt, Mary Dalrymple, and Tracy H. King, editors. 2006. *Lexical Semantics in LFG*. CSLI Publications, Stanford, CA.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James Constable and James Curran. 2009. Integrating verb-particle constructions into CCG parsing. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 114–118. Sydney, Australia.

Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. 1994. Xtag system: a wide coverage grammar for english. In *Proceedings of the 15th conference on Computational linguistics*, pages 922–928. ACL, Morristown, NJ, USA.

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

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 57–64. ACL, Morristown, NJ, USA.

Donald Hindle. 1983. User manual for fidditch, a deterministic parser. Technical Memorandum 7590-142, Naval Research Laboratory.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third Conference on Language Resources and Evaluation Conference*, pages 1974–1981. Las Palmas, Spain.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal and James R. Curran. 2007. Improving the complement/adjunct distinction in CCGBank. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 210–217. Melbourne, Australia.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The mental representation of grammatical relations*, pages 173–281. MIT Press, Cambridge, MA, USA.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Paul Martin. 2002. *The Wall Street Journal Guide to Business Style and Usage*. Free Press, New York.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *Frontiers in Corpus Annotation: Proceedings of the Workshop*, pages 24–31. Boston, MA, USA.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684–693. Hainan Island, China.

Stepan Oepen, Daniel Flickenger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. a rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2(4):575–596.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago.

Libin Shen, Lucas Champollion, and Aravind K. Joshi. 2008. LTAG-spinal and the treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. CSLI Publications, Stanford, CA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA, USA.

Daniel Tse and James R. Curran. 2008. Punctuation normalisation for cleaner treebanks and parsers. In *Proceedings of the Australian Language Technology Workshop*, volume 6, pages 151–159. ALTW, Hobart, Australia.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247. ACL, Prague, Czech Republic.

David Vadas and James R. Curran. 2008. Parsing noun phrase structure with CCG. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 335–343. ACL, Columbus, Ohio, USA.

BabelNet: Building a Very Large Multilingual Semantic Network

Roberto Navigli
Dipartimento di Informatica
Sapienza Università di Roma
navigli@di.uniroma1.it

Simone Paolo Ponzetto
Department of Computational Linguistics
Heidelberg University
ponzetto@cl.uni-heidelberg.de

Abstract

In this paper we present BabelNet – a very large, wide-coverage multilingual semantic network. The resource is automatically constructed by means of a methodology that integrates lexicographic and encyclopedic knowledge from WordNet and Wikipedia. In addition Machine Translation is also applied to enrich the resource with lexical information for all languages. We conduct experiments on new and existing gold-standard datasets to show the high quality and coverage of the resource.

1 Introduction

In many research areas of Natural Language Processing (NLP) lexical knowledge is exploited to perform tasks effectively. These include, among others, text summarization (Nastase, 2008), Named Entity Recognition (Bunescu and Paşca, 2006), Question Answering (Harabagiu et al., 2000) and text categorization (Gabrilovich and Markovitch, 2006). Recent studies in the difficult task of Word Sense Disambiguation (Navigli, 2009b, WSD) have shown the impact of the amount and quality of lexical knowledge (Cuadros and Rigau, 2006): richer knowledge sources can be of great benefit to both knowledge-lean systems (Navigli and Lapata, 2010) and supervised classifiers (Ng and Lee, 1996; Yarowsky and Florian, 2002).

Various projects have been undertaken to make lexical knowledge available in a machine readable format. A pioneering endeavor was WordNet (Fellbaum, 1998), a computational lexicon of English based on psycholinguistic theories. Subsequent projects have also tackled the significant problem of multilinguality. These include EuroWordNet (Vossen, 1998), MultiWordNet (Pianta et al., 2002), the Multilingual Central Repository (Atserias et al., 2004), and many others. However, manual construction methods inherently suffer from a number of drawbacks. First, maintaining and updating lexical knowledge resources is expensive and time-consuming. Second, such resources are typically lexicographic, and thus contain mainly concepts and only a few named entities. Third, resources for non-English languages often have a much poorer coverage since the construction effort must be repeated for every language of interest. As a result, an obvious bias exists towards conducting research in resource-rich languages, such as English.

A solution to these issues is to draw upon a large-scale collaborative resource, namely Wikipedia[1]. Wikipedia represents the perfect complement to WordNet, as it provides multilingual lexical knowledge of a mostly encyclopedic nature. While the contribution of any individual user might be imprecise or inaccurate, the continual intervention of expert contributors in all domains results in a resource of the highest quality (Giles, 2005). But while a great deal of work has been recently devoted to the automatic extraction of structured information from Wikipedia (Wu and Weld, 2007; Ponzetto and Strube, 2007; Suchanek et al., 2008; Medelyan et al., 2009, *inter alia*), the knowledge extracted is organized in a looser way than in a computational lexicon such as WordNet.

In this paper, we make a major step towards the vision of a wide-coverage multilingual knowledge resource. We present a novel methodology that produces a very large multilingual semantic network: BabelNet. This resource is created by linking Wikipedia to WordNet via an automatic mapping and by integrating lexical gaps in resource-

[1] http://download.wikipedia.org. We use the English Wikipedia database dump from November 3, 2009, which includes 3,083,466 articles. Throughout this paper, we use Sans Serif for words, SMALL CAPS for Wikipedia pages and CAPITALS for Wikipedia categories.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 216–225,
Uppsala, Sweden, 11-16 July 2010.

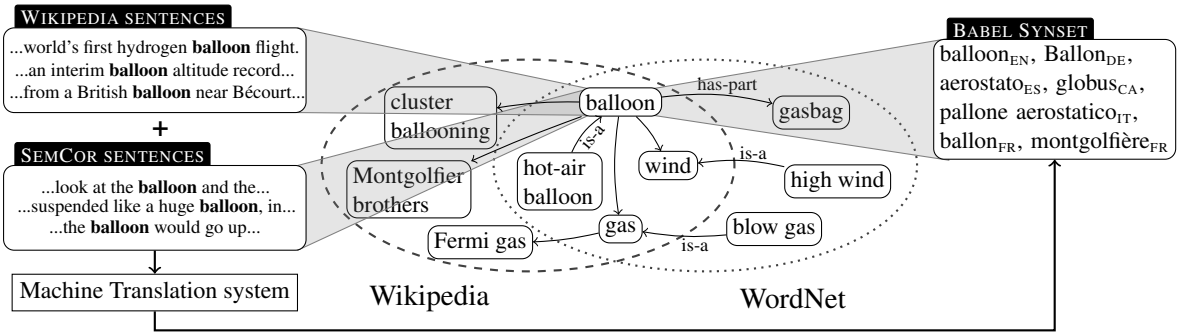


Figure 1: An illustrative overview of BabelNet.

poor languages with the aid of Machine Translation. The result is an "encyclopedic dictionary", that provides concepts and named entities lexicalized in many languages and connected with large amounts of semantic relations.

2 BabelNet

We encode knowledge as a labeled directed graph $G = (V, E)$ where V is the set of *vertices* – i.e. *concepts*[2] such as balloon – and $E \subseteq V \times R \times V$ is the set of *edges* connecting pairs of concepts. Each edge is labeled with a *semantic relation* from R, e.g. {*is-a*, *part-of*, ..., ϵ}, where ϵ denotes an unspecified semantic relation. Importantly, each vertex $v \in V$ contains a set of lexicalizations of the concept for different languages, e.g. { balloon$_{EN}$, Ballon$_{DE}$, aerostato$_{ES}$, ..., montgolfière$_{FR}$ }.

Concepts and relations in BabelNet are harvested from the largest available semantic lexicon of English, WordNet, and a wide-coverage collaboratively edited encyclopedia, the English Wikipedia (Section 3.1). We collect (a) from WordNet, all available word senses (as *concepts*) and all the semantic pointers between synsets (as *relations*); (b) from Wikipedia, all encyclopedic entries (i.e. pages, as *concepts*) and semantically unspecified *relations* from hyperlinked text.

In order to provide a *unified resource*, we merge the intersection of these two knowledge sources (i.e. their concepts in common) by establishing a mapping between Wikipedia pages and WordNet senses (Section 3.2). This avoids duplicate concepts and allows their inventories of concepts to complement each other. Finally, to enable multilinguality, we collect the lexical realizations of the available concepts in different languages by using (a) the human-generated translations provided in Wikipedia (the so-called *inter-language* links), as well as (b) a machine translation system to translate occurrences of the concepts within sense-tagged corpora, namely SemCor (Miller et al., 1993) – a corpus annotated with WordNet senses – and Wikipedia itself (Section 3.3). We call the resulting set of multilingual lexicalizations of a given concept a *babel synset*. An overview of BabelNet is given in Figure 1 (we label vertices with English lexicalizations): unlabeled edges are obtained from links in the Wikipedia pages (e.g. BALLOON (AIRCRAFT) links to WIND), whereas labeled ones from WordNet[3] (e.g. balloon^1_n *has-part* gasbag^1_n). In this paper we restrict ourselves to concepts lexicalized as nouns. Nonetheless, our methodology can be applied to all parts of speech, but in that case Wikipedia cannot be exploited, since it mainly contains nominal entities.

3 Methodology

3.1 Knowledge Resources

WordNet. The most popular lexical knowledge resource in the field of NLP is certainly WordNet, a computational lexicon of the English language. A concept in WordNet is represented as a synonym set (called *synset*), i.e. the set of words that share the same meaning. For instance, the concept wind is expressed by the following synset:

$$\{ \text{wind}^1_n, \text{air current}^1_n, \text{current of air}^1_n \},$$

where each word's subscripts and superscripts indicate their parts of speech (e.g. n stands for noun)

[2] Throughout the paper, unless otherwise stated, we use the general term *concept* to denote either a concept or a named entity.

[3] We use in the following WordNet version 3.0. We denote with w^i_p the i-th sense of a word w with part of speech p. We use word senses to unambiguously denote the corresponding synsets (e.g. plane^1_n for { airplane^1_n, aeroplane^1_n, plane^1_n }). Hereafter, we use *word sense* and *synset* interchangeably.

and sense number, respectively. For each synset, WordNet provides a textual definition, or gloss. For example, the gloss of the above synset is: "air moving from an area of high pressure to an area of low pressure".

Wikipedia. Our second resource, Wikipedia, is a Web-based collaborative encyclopedia. A Wikipedia page (henceforth, Wikipage) presents the knowledge about a specific concept (e.g. BALLOON (AIRCRAFT)) or named entity (e.g. MONTGOLFIER BROTHERS). The page typically contains hypertext linked to other relevant Wikipages. For instance, BALLOON (AIRCRAFT) is linked to WIND, GAS, and so on. The title of a Wikipage (e.g. BALLOON (AIRCRAFT)) is composed of the lemma of the concept defined (e.g. balloon) plus an optional label in parentheses which specifies its meaning if the lemma is ambiguous (e.g. AIRCRAFT VS. TOY). Wikipages also provide inter-language links to their counterparts in other languages (e.g. BALLOON (AIRCRAFT) links to the Spanish page AEROSTATO). Finally, some Wikipages are redirections to other pages, e.g. the Spanish BALÓN AEROSTÁTICO redirects to AEROSTATO.

3.2 Mapping Wikipedia to WordNet

The first phase of our methodology aims to establish links between Wikipages and WordNet senses. We aim to acquire a mapping μ such that, for each Wikipage w, we have:

$$\mu(w) = \begin{cases} s \in Senses_{\text{WN}}(w) & \text{if a link can be established,} \\ \epsilon & \text{otherwise,} \end{cases}$$

where $Senses_{\text{WN}}(w)$ is the set of senses of the *lemma* of w in WordNet. For example, if our mapping methodology linked BALLOON (AIRCRAFT) to the corresponding WordNet sense balloon^1_n, we would have μ(BALLOON (AIRCRAFT)) = balloon^1_n.

In order to establish a mapping between the two resources, we first identify the disambiguation contexts for Wikipages (Section 3.2.1) and WordNet senses (Section 3.2.2). Next, we intersect these contexts to perform the mapping (see Section 3.2.3).

3.2.1 Disambiguation Context of a Wikipage

Given a Wikipage w, we use the following information as disambiguation context:

- **Sense labels**: e.g. given the page BALLOON (AIRCRAFT), the word aircraft is added to the disambiguation context.
- **Links**: the titles' lemmas of the pages linked from the target Wikipage (i.e., outgoing links). For instance, the links in the Wikipage BALLOON (AIRCRAFT) include wind, gas, etc.
- **Categories**: Wikipages are typically classified according to one or more categories. For example, the Wikipage BALLOON (AIRCRAFT) is categorized as BALLOONS, BALLOONING, etc. While many categories are very specific and do not appear in WordNet (e.g., SWEDISH WRITERS or SCIENTISTS WHO COMMITTED SUICIDE), we use their syntactic heads as disambiguation context (i.e. writer and scientist, respectively).

Given a Wikipage w, we define its disambiguation context $Ctx(w)$ as the set of words obtained from all of the three sources above.

3.2.2 Disambiguation Context of a WordNet Sense

Given a WordNet sense s and its synset S, we collect the following information:

- **Synonymy**: all synonyms of s in S. For instance, given the sense airplane^1_n and its corresponding synset { airplane^1_n, aeroplane^1_n, plane^1_n }, the words contained therein are included in the context.
- **Hypernymy/Hyponymy**: all synonyms in the synsets H such that H is either a hypernym (i.e., a generalization) or a hyponym (i.e., a specialization) of S. For example, given balloon^1_n, we include the words from its hypernym { $\text{lighter-than-air craft}^1_n$ } and all its hyponyms (e.g. { $\text{hot-air balloon}^1_n$ }).
- **Sisterhood**: words from the sisters of S. A sister synset S' is such that S and S' have a common direct hypernym. For example, given balloon^1_n, it can be found that { balloon^1_n } and { airship^1_n, dirigible^1_n } are sisters. Thus airship and dirigible are included in the disambiguation context of s.
- **Gloss**: the set of lemmas of the content words occurring within the WordNet gloss of S.

We thus define the disambiguation context $Ctx(s)$ of sense s as the set of words obtained from all of the four sources above.

3.2.3 Mapping Algorithm

In order to link each Wikipedia page to a WordNet sense, we perform the following steps:

- Initially, our mapping μ is empty, i.e. it links each Wikipage w to ϵ.
- For each Wikipage w whose lemma is monosemous both in Wikipedia and WordNet we map w to its only WordNet sense.
- For each remaining Wikipage w for which no mapping was previously found (i.e., $\mu(w) = \epsilon$), we assign the most likely sense to w based on the maximization of the conditional probabilities $p(s|w)$ over the senses $s \in Senses_{\text{WN}}(w)$ (no mapping is established if a tie occurs).

To find the mapping of a Wikipage w, we need to compute the conditional probability $p(s|w)$ of selecting the WordNet sense s given w. The sense s which maximizes this probability is determined as follows:

$$\mu(w) = \underset{s \in Senses_{\text{WN}}(w)}{\operatorname{argmax}} p(s|w) = \underset{s}{\operatorname{argmax}} \frac{p(s,w)}{p(w)} = \underset{s}{\operatorname{argmax}}\, p(s,w)$$

The latter formula is obtained by observing that $p(w)$ does not influence our maximization, as it is a constant independent of s. As a result, determining the most appropriate sense s consists of finding the sense s that maximizes the joint probability $p(s,w)$. We estimate $p(s,w)$ as:

$$p(s,w) = \frac{score(s,w)}{\sum_{\substack{s' \in Senses_{\text{WN}}(w),\\ w' \in Senses_{\text{Wiki}}(w)}} score(s',w')},$$

where $score(s,w) = |Ctx(s) \cap Ctx(w)| + 1$ (we add 1 as a smoothing factor). Thus, in our algorithm we determine the best sense s by computing the intersection of the disambiguation contexts of s and w, and normalizing by the scores summed over all senses of w in Wikipedia and WordNet. More details on the mapping algorithm can be found in Ponzetto and Navigli (2010).

3.3 Translating Babel Synsets

So far we have linked English Wikipages to WordNet senses. Given a Wikipage w, and provided it is mapped to a sense s (i.e., $\mu(w) = s$), we create a babel synset $S \cup W$, where S is the WordNet synset to which sense s belongs, and W includes: (i) w; (ii) all its inter-language links (that is, translations of the Wikipage to other languages); (iii) the redirections to the inter-language links found in the Wikipedia of the target language. For instance, given that μ(BALLOON) = balloon^1_n, the corresponding babel synset is { balloon$_{\text{EN}}$, Ballon$_{\text{DE}}$, aerostato$_{\text{ES}}$, balón aerostático$_{\text{ES}}$, ..., pallone aerostatico$_{\text{IT}}$ }. However, two issues arise: first, a concept might be covered only in one of the two resources (either WordNet or Wikipedia), meaning that no link can be established (e.g., FERMI GAS or gasbag^1_n in Figure 1); second, even if covered in both resources, the Wikipage for the concept might not provide any translation for the language of interest (e.g., the Catalan for BALLOON is missing in Wikipedia).

In order to address the above issues and thus guarantee high coverage for all languages we developed a methodology for translating senses in the babel synset to missing languages. Given a WordNet word sense in our babel synset of interest (e.g. balloon^1_n) we collect its occurrences in SemCor (Miller et al., 1993), a corpus of more than 200,000 words annotated with WordNet senses. We do the same for Wikipages by retrieving sentences in Wikipedia with links to the Wikipage of interest. By repeating this step for each English lexicalization in a babel synset, we obtain a collection of sentences for the babel synset (see left part of Figure 1). Next, we apply state-of-the-art Machine Translation[4] and translate the set of sentences in all the languages of interest. Given a specific term in the initial babel synset, we collect the set of its translations. We then identify the most frequent translation in each language and add it to the babel synset. Note that translations are sense-specific, as the context in which a term occurs is provided to the translation system.

3.4 Example

We now illustrate the execution of our methodology by way of an example. Let us focus on the Wikipage BALLOON (AIRCRAFT). The word is polysemous both in Wikipedia and WordNet. In the first phase of our methodology we aim to find a mapping μ(BALLOON (AIRCRAFT)) to an appropriate WordNet sense of the word. To

[4] We use the Google Translate API. An initial prototype used a statistical machine translation system based on Moses (Koehn et al., 2007) and trained on Europarl (Koehn, 2005). However, we found such system unable to cope with many technical names, such as in the domains of sciences, literature, history, etc.

this end we construct the disambiguation context for the Wikipage by including words from its label, links and categories (cf. Section 3.2.1). The context thus includes, among others, the following words: aircraft, wind, airship, lighter-than-air. We now construct the disambiguation context for the two WordNet senses of balloon (cf. Section 3.2.2), namely the aircraft (#1) and the toy (#2) senses. To do so, we include words from their synsets, hypernyms, hyponyms, sisters, and glosses. The context for balloon^1_n includes: aircraft, craft, airship, lighter-than-air. The context for balloon^2_n contains: toy, doll, hobby. The sense with the largest intersection is #1, so the following mapping is established: μ(BALLOON (AIRCRAFT)) = balloon^1_n. After the first phase, our babel synset includes the following English words from WordNet plus the Wikipedia inter-language links to other languages (we report German, Spanish and Italian): { $\text{balloon}_{\text{EN}}$, $\text{Ballon}_{\text{DE}}$, $\text{aerostato}_{\text{ES}}$, $\text{balón aerostático}_{\text{ES}}$, $\text{pallone aerostatico}_{\text{IT}}$ }.

In the second phase (see Section 3.3), we collect all the sentences in SemCor and Wikipedia in which the above English word sense occurs. We translate these sentences with the Google Translate API and select the most frequent translation in each language. As a result, we can enrich the initial babel synset with the following words: $\text{mongolfière}_{\text{FR}}$, $\text{globus}_{\text{CA}}$, globo_{ES}, $\text{mongolfiera}_{\text{IT}}$. Note that we had no translation for Catalan and French in the first phase, because the inter-language link was not available, and we also obtain new lexicalizations for the Spanish and Italian languages.

4 Experiment 1: Mapping Evaluation

Experimental setting. We first performed an evaluation of the quality of our mapping from Wikipedia to WordNet. To create a gold standard for evaluation we considered all lemmas whose senses are contained both in WordNet and Wikipedia: the intersection between the two resources contains 80,295 lemmas which correspond to 105,797 WordNet senses and 199,735 Wikipedia pages. The average polysemy is 1.3 and 2.5 for WordNet senses and Wikipages, respectively (2.8 and 4.7 when excluding monosemous words). We then selected a random sample of 1,000 Wikipages and asked an annotator with previous experience in lexicographic annotation to provide the correct WordNet sense for each page (an empty sense label was given, if no correct mapping was possible). The gold-standard dataset includes 505 non-empty mappings, i.e. Wikipages with a corresponding WordNet sense. In order to quantify the quality of the annotations and the difficulty of the task, a second annotator sense tagged a subset of 200 pages from the original sample. Our annotators achieved a κ inter-annotator agreement (Carletta, 1996) of 0.9, indicating almost perfect agreement.

	P	R	F_1	A
Mapping algorithm	81.9	77.5	79.6	84.4
MFS BL	24.3	47.8	32.2	24.3
Random BL	23.8	46.8	31.6	23.9

Table 1: Performance of the mapping algorithm.

Results and discussion. Table 1 summarizes the performance of our mapping algorithm against the manually annotated dataset. Evaluation is performed in terms of standard measures of precision, recall, and F_1-measure. In addition we calculate accuracy, which also takes into account empty sense labels. As baselines we use the most frequent WordNet sense (MFS), and a random sense assignment.

The results show that our method achieves almost 80% F_1 and it improves over the baselines by a large margin. The final mapping contains 81,533 pairs of Wikipages and word senses they map to, covering 55.7% of the noun senses in WordNet. As for the baselines, the most frequent sense is just 0.6% and 0.4% above the random baseline in terms of F_1 and accuracy, respectively. A χ^2 test reveals in fact no statistical significant difference at $p < 0.05$. This is related to the random distribution of senses in our dataset and the Wikipedia unbiased coverage of WordNet senses. So selecting the first WordNet sense rather than any other sense for each target page represents a choice as arbitrary as picking a sense at random.

5 Experiment 2: Translation Evaluation

We perform a second set of experiments concerning the quality of the acquired concepts. This is assessed in terms of coverage against gold-standard resources (Section 5.1) and against a manually-validated dataset of translations (Section 5.2).

Language	Word senses	Synsets
German	15,762	9,877
Spanish	83,114	55,365
Catalan	64,171	40,466
Italian	57,255	32,156
French	44,265	31,742

Table 2: Size of the gold-standard wordnets.

5.1 Automatic Evaluation

Datasets. We compare BabelNet against gold-standard resources for 5 languages, namely: the subset of GermaNet (Lemnitzer and Kunze, 2002) included in EuroWordNet for German, MultiWordNet (Pianta et al., 2002) for Italian, the Multilingual Central Repository for Spanish and Catalan (Atserias et al., 2004), and WOrdnet Libre du Français (Benoît and Fišer, 2008, WOLF) for French. In Table 2 we report the number of synsets and word senses available in the gold-standard resources for the 5 languages.

Measures. Let $\mathcal{B}$ be BabelNet, $\mathcal{F}$ our gold-standard non-English wordnet (e.g. GermaNet), and let $\mathcal{E}$ be the English WordNet. All the gold-standard non-English resources, as well as BabelNet, are linked to the English WordNet: given a synset $S_{\mathcal{F}} \in \mathcal{F}$, we denote its corresponding babel synset as $S_{\mathcal{B}}$ and its synset in the English WordNet as $S_{\mathcal{E}}$. We assess the coverage of BabelNet against our gold-standard wordnets both in terms of synsets and word senses. For synsets, we calculate coverage as follows:

$$\text{SynsetCov}(\mathcal{B},\mathcal{F}) = \frac{\sum_{S_{\mathcal{F}}\in\mathcal{F}} \delta(S_{\mathcal{B}}, S_{\mathcal{F}})}{|\{S_{\mathcal{F}} \in \mathcal{F}\}|},$$

where $\delta(S_{\mathcal{B}}, S_{\mathcal{F}}) = 1$ if the two synsets $S_{\mathcal{B}}$ and $S_{\mathcal{F}}$ have a synonym in common, 0 otherwise. That is, synset coverage is determined as the percentage of synsets of $\mathcal{F}$ that share a term with the corresponding babel synsets. For word senses we calculate a similar measure of coverage:

$$\text{WordCov}(\mathcal{B},\mathcal{F}) = \frac{\sum_{S_{\mathcal{F}}\in\mathcal{F}} \sum_{s_{\mathcal{F}}\in S_{\mathcal{F}}} \delta'(s_{\mathcal{F}}, S_{\mathcal{B}})}{|\{s_{\mathcal{F}} \in S_{\mathcal{F}} : S_{\mathcal{F}} \in \mathcal{F}\}|},$$

where $s_{\mathcal{F}}$ is a word sense in synset $S_{\mathcal{F}}$ and $\delta'(s_{\mathcal{F}}, S_{\mathcal{B}}) = 1$ if $s_{\mathcal{F}} \in S_{\mathcal{B}}$, 0 otherwise. That is we calculate the ratio of word senses in our gold-standard resource $\mathcal{F}$ that also occur in the corresponding synset $S_{\mathcal{B}}$ to the overall number of senses in $\mathcal{F}$.

However, our gold-standard resources cover only a portion of the English WordNet, whereas the overall coverage of BabelNet is much higher. We calculate extra coverage for synsets as follows:

$$\text{SynsetExtraCov}(\mathcal{B},\mathcal{F}) = \frac{\sum_{S_{\mathcal{E}}\in\mathcal{E}\setminus\mathcal{F}} \delta(S_{\mathcal{B}}, S_{\mathcal{E}})}{|\{S_{\mathcal{F}} \in \mathcal{F}\}|}.$$

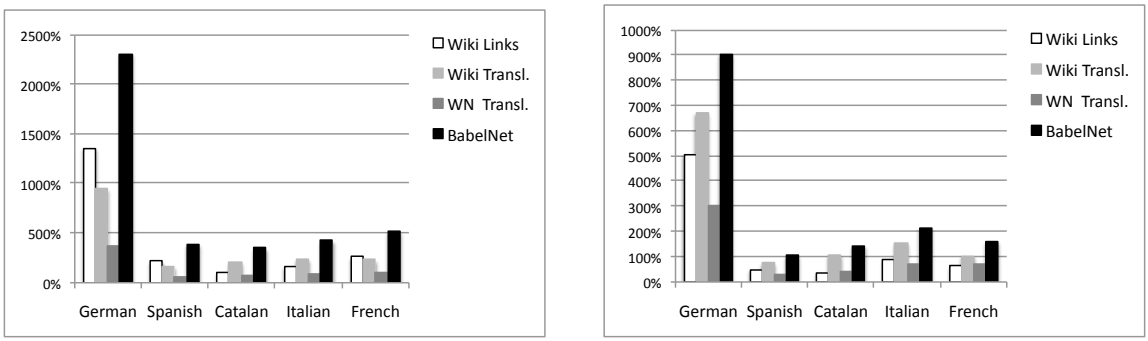
Similarly, we calculate extra coverage for word senses in BabelNet corresponding to WordNet synsets not covered by the reference resource $\mathcal{F}$.

Results and discussion. We evaluate the coverage and extra coverage of word senses and synsets at different stages: (a) using only the inter-language links from Wikipedia (WIKI Links); (b) and (c) using only the automatic translations of the sentences from Wikipedia (WIKI Transl.) or SemCor (WN Transl.); (d) using all available translations, i.e. BABELNET.

Coverage results are reported in Table 3. The percentage of word senses covered by BabelNet ranges from 52.9% (Italian) to 66.4 (Spanish) and 86.0% (French). Synset coverage ranges from 73.3% (Catalan) to 76.6% (Spanish) and 92.9% (French). As expected, synset coverage is higher, because a synset in the reference resource is considered to be covered if it shares at least one word with the corresponding synset in BabelNet.

Numbers for the extra coverage, which provides information about the percentage of word senses and synsets in BabelNet but not in the gold-standard resources, are given in Figure 2. The results show that we provide for all languages a high extra coverage for both word senses – between 340.1% (Catalan) and 2,298% (German) – and synsets – between 102.8% (Spanish) and 902.6% (German).

Table 3 and Figure 2 show that the best results are obtained when combining *all* available translations, i.e. both from Wikipedia and the machine translation system. The performance figures suffer from the errors of the mapping phase (see Section 4). Nonetheless, the results are generally high, with a peak for French, since WOLF has been created semi-automatically by combining several resources, *including* Wikipedia. The relatively low word sense coverage for Italian (55.4%) is, instead, due to the lack of many common words in the Italian gold-standard synsets. Examples include whip$_{\text{EN}}$ translated as staffile$_{\text{IT}}$ but not as the more common frusta$_{\text{IT}}$, playboy$_{\text{EN}}$ translated as vitaiolo$_{\text{IT}}$ but not gigolò$_{\text{IT}}$, etc.



(a) word senses

(b) synsets

Figure 2: Extra coverage against gold-standard wordnets: word senses (a) and synsets (b).

	Resource	Method	SENSES	SYNSETS
German	WIKI	Links	39.6	50.7
		Transl.	42.6	58.2
	WN	Transl.	21.0	28.6
	BABELNET	All	**57.6**	**73.4**
Spanish	WIKI	Links	34.4	40.7
		Transl.	47.9	56.1
	WN	Transl.	25.2	30.0
	BABELNET	All	**66.4**	**76.6**
Catalan	WIKI	Links	20.3	25.2
		Transl.	46.9	54.1
	WN	Transl.	25.0	29.6
	BABELNET	All	**64.0**	**73.3**
Italian	WIKI	Links	28.1	40.0
		Transl.	39.9	58.0
	WN	Transl.	19.7	28.7
	BABELNET	All	**52.9**	**73.7**
French	WIKI	Links	70.0	72.4
		Transl.	69.6	79.6
	WN	Transl.	16.3	19.4
	BABELNET	All	**86.0**	**92.9**

Table 3: Coverage against gold-standard wordnets (we report percentages).

5.2 Manual Evaluation

Experimental setup. The automatic evaluation quantifies how much of the gold-standard resources is covered by BabelNet. However, it does not say anything about the precision of the *additional* lexicalizations provided by BabelNet. Given that our resource has displayed a remarkably high extra coverage – ranging from 340% to 2,298% of the national wordnets (see Figure 2) – we performed a second evaluation to assess its precision. For each of our 5 languages, we selected a random set of 600 babel synsets composed as follows: 200 synsets whose senses exist in WordNet only, 200 synsets in the intersection between WordNet and Wikipedia (i.e. those mapped with our method illustrated in Section 3.2), 200 synsets whose lexicalizations exist in Wikipedia only. Therefore, our dataset included $600 \times 5 = 3{,}000$ babel synsets. None of the synsets was covered by any of the five reference wordnets. The babel synsets were manually validated by expert annotators who decided which senses (i.e. lexicalizations) were appropriate given the corresponding WordNet gloss and/or Wikipage.

Results and discussion. We report the results in Table 4. For each language (rows) and for each of the three regions of BabelNet (columns), we report precision (i.e. the percentage of synonyms deemed correct) and, in parentheses, the overall number of synonyms evaluated. The results show that the different regions of BabelNet contain translations of different quality: while on average translations for WordNet-only synsets have a precision around 72%, when Wikipedia comes into play the performance increases considerably (around 80% in the intersection and 95% with Wikipedia-only translations). As can be seen from the figures in parentheses, the number of translations available in the presence of Wikipedia is higher. This quantitative difference is due to our method collecting many translations from the redirections in the Wikipedia of the target language (Section 3.3), as well as to the paucity of examples in SemCor for many synsets. In addition, some of the synsets in WordNet with no Wikipedia counterpart are very difficult to translate. Examples include terms like stammel, crape fern, baseball clinic, and many others for which we could

Language	WN	WN ∩ Wiki	Wiki
German	73.76 (282)	78.37 (777)	97.74 (709)
Spanish	69.45 (275)	78.53 (643)	92.46 (703)
Catalan	75.58 (258)	82.98 (517)	92.71 (398)
Italian	72.32 (271)	80.83 (574)	99.09 (552)
French	67.16 (268)	77.43 (709)	96.44 (758)

Table 4: Precision of BabelNet on synonyms in WordNet (WN), Wikipedia (Wiki) and their intersection (WN ∩ Wiki): percentage and total number of words (in parentheses) are reported.

not find translations in major editions of bilingual dictionaries. In contrast, good translations were produced using our machine translation method when enough sentences were available. Examples are: chaudrée de poisson$_{\text{FR}}$ for fish chowder$_{\text{EN}}$, grano de café$_{\text{ES}}$ for coffee bean$_{\text{EN}}$, etc.

6 Related Work

Previous attempts to manually build multilingual resources have led to the creation of a multitude of wordnets such as EuroWordNet (Vossen, 1998), MultiWordNet (Pianta et al., 2002), BalkaNet (Tufiş et al., 2004), Arabic WordNet (Black et al., 2006), the Multilingual Central Repository (Atserias et al., 2004), bilingual electronic dictionaries such as EDR (Yokoi, 1995), and fully-fledged frameworks for the development of multilingual lexicons (Lenci et al., 2000). As it is often the case with manually assembled resources, these lexical knowledge repositories are hindered by high development costs and an insufficient coverage. This barrier has led to proposals that acquire multilingual lexicons from either parallel text (Gale and Church, 1993; Fung, 1995, *inter alia*) or monolingual corpora (Sammer and Soderland, 2007; Haghighi et al., 2008). The disambiguation of bilingual dictionary glosses has also been proposed to create a bilingual semantic network from a machine readable dictionary (Navigli, 2009a). Recently, Etzioni et al. (2007) and Mausam et al. (2009) presented methods to produce massive multilingual translation dictionaries from Web resources such as online lexicons and Wiktionaries. However, while providing lexical resources on a very large scale for hundreds of thousands of language pairs, these do not encode semantic relations between concepts denoted by their lexical entries.

The research closest to ours is presented by de Melo and Weikum (2009), who developed a Universal WordNet (UWN) by automatically acquiring a semantic network for languages other than English. UWN is bootstrapped from WordNet and is built by collecting evidence extracted from existing wordnets, translation dictionaries, and parallel corpora. The result is a graph containing 800,000 words from over 200 languages in a hierarchically structured semantic network with over 1.5 million links from words to word senses. Our work goes one step further by (1) developing an even larger multilingual resource including *both* lexical semantic and encyclopedic knowledge, (2) enriching the structure of the 'core' semantic network (i.e. the semantic pointers from WordNet) with topical, semantically unspecified relations from the link structure of Wikipedia. This result is essentially achieved by complementing WordNet with Wikipedia, as well as by leveraging the multilingual structure of the latter. Previous attempts at linking the two resources have been proposed. These include associating Wikipedia pages with the most frequent WordNet sense (Suchanek et al., 2008), extracting domain information from Wikipedia and providing a manual mapping to WordNet concepts (Auer et al., 2007), a model based on vector spaces (Ruiz-Casado et al., 2005), a supervised approach using keyword extraction (Reiter et al., 2008), as well as automatically linking Wikipedia categories to WordNet based on structural information (Ponzetto and Navigli, 2009). In contrast to previous work, BabelNet is the first proposal that integrates the relational structure of WordNet with the semi-structured information from Wikipedia into a unified, wide-coverage, multilingual semantic network.

7 Conclusions

In this paper we have presented a novel methodology for the automatic construction of a large multilingual lexical knowledge resource. Key to our approach is the establishment of a mapping between a multilingual encyclopedic knowledge repository (Wikipedia) and a computational lexicon of English (WordNet). This integration process has several advantages. Firstly, the two resources contribute different kinds of lexical knowledge, one is concerned mostly with named entities, the other with concepts. Secondly, while Wikipedia is less structured than WordNet, it provides large

amounts of semantic relations and can be leveraged to enable multilinguality. Thus, even when they overlap, the two resources provide complementary information about the same named entities or concepts. Further, we contribute a large set of sense occurrences harvested from Wikipedia and SemCor, a corpus that we input to a state-of-the-art machine translation system to fill in the gap between resource-rich languages – such as English – and resource-poorer ones. Our hope is that the availability of such a language-rich resource[5] will enable many non-English and multilingual NLP applications to be developed.

Our experiments show that our fully-automated approach produces a large-scale lexical resource with high accuracy. The resource includes millions of semantic relations, mainly from Wikipedia (however, WordNet relations are labeled), and contains almost 3 million concepts (6.7 labels per concept on average). As pointed out in Section 5, such coverage is much wider than that of existing wordnets in non-English languages. While BabelNet currently includes 6 languages, links to freely-available wordnets[6] can immediately be established by utilizing the English WordNet as an interlanguage index. Indeed, BabelNet can be extended to virtually any language of interest. In fact, our translation method allows it to cope with any resource-poor language.

As future work, we plan to apply our method to other languages, including Eastern European, Arabic, and Asian languages. We also intend to link missing concepts in WordNet, by establishing their most likely hypernyms – e.g., à la Snow et al. (2006). We will perform a semi-automatic validation of BabelNet, e.g. by exploiting Amazon's Mechanical Turk (Callison-Burch, 2009) or designing a collaborative game (von Ahn, 2006) to validate low-ranking mappings and translations. Finally, we aim to apply BabelNet to a variety of applications which are known to benefit from a wide-coverage knowledge resource. We have already shown that the English-only subset of BabelNet allows simple knowledge-based algorithms to compete with supervised systems in standard coarse-grained and domain-specific WSD settings (Ponzetto and Navigli, 2010). We plan in the near future to apply BabelNet to the challenging task of cross-lingual WSD (Lefever and Hoste, 2009).

[5] BabelNet can be freely downloaded for research purposes at `http://lcl.uniroma1.it/babelnet`.

[6] `http://www.globalwordnet.org`.

References

Jordi Atserias, Luis Villarejo, German Rigau, Eneko Agirre, John Carroll, Bernardo Magnini, and Piek Vossen. 2004. The MEANING multilingual central repository. In *Proc. of GWC-04*, pages 80–210.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ive. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of 6th International Semantic Web Conference joint with 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, pages 722–735.

Sagot Benoît and Darja Fišer. 2008. Building a free French WordNet from multilingual resources. In *Proceedings of the Ontolex 2008 Workshop*.

William Black, Sabri Elkateb Horacio Rodriguez, Musa Alkhalifa, Piek Vossen, and Adam Pease. 2006. Introducing the Arabic WordNet project. In *Proc. of GWC-06*, pages 295–299.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, pages 9–16.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proc. of EMNLP-09*, pages 286–295.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Montse Cuadros and German Rigau. 2006. Quality assessment of large scale knowledge resources. In *Proc. of EMNLP-06*, pages 534–541.

Gerard de Melo and Gerhard Weikum. 2009. Towards a universal wordnet by learning from combined evidence. In *Proc. of CIKM-09*, pages 513–522.

Oren Etzioni, Kobi Reiter, Stephen Soderland, and Marcus Sammer. 2007. Lexical translation with application to image search on the Web. In *Proceedings of Machine Translation Summit XI*.

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

Pascale Fung. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proc. of ACL-95*, pages 236–243.

Evgeniy Gabrilovich and Shaul Markovitch. 2006. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proc. of AAAI-06*, pages 1301–1306.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102.

Jim Giles. 2005. Internet encyclopedias go head to head. *Nature*, 438:900–901.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL-08*, pages 771–779.

Sanda M. Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *Proc. of TREC-9*, pages 479–488.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Comp. Vol. to Proc. of ACL-07*, pages 177–180.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*.

Els Lefever and Veronique Hoste. 2009. Semeval-2010 task 3: Cross-lingual Word Sense Disambiguation. In *Proc. of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 82–87, Boulder, Colorado.

Lothar Lemnitzer and Claudia Kunze. 2002. GermaNet – representation, visualization, application. In *Proc. of LREC '02*, pages 1485–1491.

Alessandro Lenci, Nuria Bel, Federica Busa, Nicoletta Calzolari, Elisabetta Gola, Monica Monachini, Antoine Ogonowski, Ivonne Peters, Wim Peters, Nilda Ruimy, Marta Villegas, and Antonio Zampolli. 2000. SIMPLE: A general framework for the development of multilingual lexicons. *International Journal of Lexicography*, 13(4):249–263.

Mausam, Stephen Soderland, Oren Etzioni, Daniel Weld, Michael Skinner, and Jeff Bilmes. 2009. Compiling a massive, multilingual dictionary via probabilistic inference. In *Proc. of ACL-IJCNLP-09*, pages 262–270.

Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from Wikipedia. *Int. J. Hum.-Comput. Stud.*, 67(9):716–754.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, N.J.

Vivi Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and activation spreading. In *Proc. of EMNLP-08*, pages 763–772.

Roberto Navigli and Mirella Lapata. 2010. An experimental study on graph connectivity for unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Anaylsis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli. 2009a. Using cycles and quasi-cycles to disambiguate dictionary glosses. In *Proc. of EACL-09*, pages 594–602.

Roberto Navigli. 2009b. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach. In *Proc. of ACL-96*, pages 40–47.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: Developing an aligned multilingual database. In *Proc. of GWC-02*, pages 21–25.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proc. of IJCAI-09*, pages 2083–2088.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich Word Sense Disambiguation rivaling supervised system. In *Proc. of ACL-10*.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proc. of AAAI-07*, pages 1440–1445.

Nils Reiter, Matthias Hartung, and Anette Frank. 2008. A resource-poor approach for linking ontology classes to Wikipedia articles. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing*, volume 1 of *Research in Computational Semantics*, pages 381–387. College Publications, London, England.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of Wikipedia encyclopedic entries to WordNet synsets. In *Advances in Web Intelligence*, volume 3528 of *Lecture Notes in Computer Science*. Springer Verlag.

Marcus Sammer and Stephen Soderland. 2007. Building a sense-distinguished multilingual lexicon from monolingual corpora and bilingual lexicons. In *Proceedings of Machine Translation Summit XI*.

Rion Snow, Dan Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proc. of COLING-ACL-06*, pages 801–808.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Dan Tufiş, Dan Cristea, and Sofia Stamou. 2004. BalkaNet: Aims, methods, results and perspectives. a general overview. *Romanian Journal on Science and Technology of Information*, 7(1-2):9–43.

Luis von Ahn. 2006. Games with a purpose. *IEEE Computer*, 6(39):92–94.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer, Dordrecht, The Netherlands.

Fei Wu and Daniel Weld. 2007. Automatically semantifying Wikipedia. In *Proc. of CIKM-07*, pages 41–50.

David Yarowsky and Radu Florian. 2002. Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering*, 9(4):293–310.

Toshio Yokoi. 1995. The EDR electronic dictionary. *Communications of the ACM*, 38(11):42–44.

Fully Unsupervised Core-Adjunct Argument Classification

Omri Abend*
Institute of Computer Science
The Hebrew University
omria01@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University
arir@cs.huji.ac.il

Abstract

The core-adjunct argument distinction is a basic one in the theory of argument structure. The task of distinguishing between the two has strong relations to various basic NLP tasks such as syntactic parsing, semantic role labeling and subcategorization acquisition. This paper presents a novel unsupervised algorithm for the task that uses no supervised models, utilizing instead state-of-the-art syntactic induction algorithms. This is the first work to tackle this task in a fully unsupervised scenario.

1 Introduction

The distinction between core arguments (henceforth, cores) and adjuncts is included in most theories on argument structure (Dowty, 2000). The distinction can be viewed syntactically, as one between obligatory and optional arguments, or semantically, as one between arguments whose meanings are predicate dependent and independent. The latter (cores) are those whose function in the described event is to a large extent determined by the predicate, and are obligatory. Adjuncts are optional arguments which, like adverbs, modify the meaning of the described event in a predictable or predicate-independent manner.

Consider the following examples:

1. The surgeon operated [on his colleague].
2. Ron will drop by [after lunch].
3. Yuri played football [in the park].

The marked argument is a core in 1 and an adjunct in 2 and 3. Adjuncts form an independent semantic unit and their semantic role can often be inferred independently of the predicate (e.g., [after lunch] is usually a temporal modifier). Core roles are more predicate-specific, e.g., [on his colleague] has a different meaning with the verbs 'operate' and 'count'.

Sometimes the same argument plays a different role in different sentences. In (3), [in the park] places a well-defined situation (Yuri playing football) in a certain location. However, in "The troops are based [in the park]", the same argument is obligatory, since being based requires a place to be based in.

Distinguishing between the two argument types has been discussed extensively in various formulations in the NLP literature, notably in PP attachment, semantic role labeling (SRL) and subcategorization acquisition. However, no work has tackled it yet in a fully unsupervised scenario. Unsupervised models reduce reliance on the costly and error prone manual multi-layer annotation (POS tagging, parsing, core-adjunct tagging) commonly used for this task. They also allow to examine the nature of the distinction and to what extent it is accounted for in real data in a theory-independent manner.

In this paper we present a fully unsupervised algorithm for core-adjunct classification. We utilize leading fully unsupervised grammar induction and POS induction algorithms. We focus on prepositional arguments, since non-prepositional ones are generally cores. The algorithm uses three measures based on different characterizations of the core-adjunct distinction, and combines them using an ensemble method followed by self-training. The measures used are based on selectional preference, predicate-slot collocation and argument-slot collocation.

We evaluate against PropBank (Palmer et al., 2005), obtaining roughly 70% accuracy when evaluated on the prepositional arguments and more than 80% for the entire argument set. These results are substantially better than those obtained by a non-trivial baseline.

* Omri Abend is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 226–236,
Uppsala, Sweden, 11-16 July 2010.

Section 2 discusses the core-adjunct distinction. Section 3 describes the algorithm. Sections 4 and 5 present our experimental setup and results.

2 Core-Adjunct in Previous Work

PropBank. PropBank (PB) (Palmer et al., 2005) is a widely used corpus, providing SRL annotation for the entire WSJ Penn Treebank. Its core labels are predicate specific, while adjunct (or modifiers under their terminology) labels are shared across predicates. The adjuncts are subcategorized into several classes, the most frequent of which are locative, temporal and manner[1].

The organization of PropBank is based on the notion of diathesis alternations, which are (roughly) defined to be alternations between two subcategorization frames that preserve meaning or change it systematically. The frames in which each verb appears were collected and sets of alternating frames were defined. Each such set was assumed to have a unique set of roles, named 'role-set'. These roles include all roles appearing in any of the frames, except of those defined as adjuncts.

Adjuncts are defined to be optional arguments appearing with a wide variety of verbs and frames. They can be viewed as fixed points with respect to alternations, i.e., as arguments that do not change their place or slot when the frame undergoes an alternation. This follows the notions of optionality and compositionality that define adjuncts.

Detecting diathesis alternations automatically is difficult (McCarthy, 2001), requiring an initial acquisition of a subcategorization lexicon. This alone is a challenging task tackled in the past using supervised parsers (see below).

FrameNet. FrameNet (FN) (Baker et al., 1998) is a large-scale lexicon based on frame semantics. It takes a different approach from PB to semantic roles. Like PB, it distinguishes between core and non-core arguments, but it does so for each and every frame separately. It does not commit that a semantic role is consistently tagged as a core or a non-core across frames. For example, the semantic role 'path' is considered core in the 'Self Motion' frame, but as non-core in the 'Placing' frame. Another difference is that FN does not allow any type of non-core argument to attach to a given frame. For instance, while the 'Getting' frame allows a 'Duration' non-core argument, the 'Active Perception' frame does not.

PB and FN tend to agree in clear (prototypical) cases, but to differ in others. For instance, both schemes would tag "Yuri played football [in the park]" as an adjunct and "The commander placed a guard [in the park]" as a core. However, in "He walked [into his office]", the marked argument is tagged as a directional adjunct in PB but as a 'Direction' core in FN.

Under both schemes, non-cores are usually confined to a few specific semantic domains, notably time, place and manner, in contrast to cores that are not restricted in their scope of applicability. This approach is quite common, e.g., the COBUILD English grammar (Willis, 2004) categorizes adjuncts to be of manner, aspect, opinion, place, time, frequency, duration, degree, extent, emphasis, focus and probability.

Semantic Role Labeling. Work in SRL does not tackle the core-adjunct task separately but as part of general argument classification. Supervised approaches obtain an almost perfect score in distinguishing between the two in an in-domain scenario. For instance, the confusion matrix in (Toutanova et al., 2008) indicates that their model scores 99.5% accuracy on this task. However, adaptation results are lower, with the best two models in the CoNLL 2005 shared task (Carreras and Màrquez, 2005) achieving 95.3% (Pradhan et al., 2008) and 95.6% (Punyakanok et al., 2008) accuracy in an adaptation between the relatively similar corpora WSJ and Brown.

Despite the high performance in supervised scenarios, tackling the task in an unsupervised manner is not easy. The success of supervised methods stems from the fact that the predicate-slot combination (slot is represented in this paper by its preposition) strongly determines whether a given argument is an adjunct or a core (see Section 3.4). Supervised models are provided with an annotated corpus from which they can easily learn the mapping between predicate-slot pairs and their core/adjunct label. However, induction of the mapping in an unsupervised manner must be based on inherent core-adjunct properties. In addition, supervised models utilize supervised parsers and POS taggers, while the current state-of-the-art in unsupervised parsing and POS tagging is considerably worse than their supervised counterparts.

This challenge has some resemblance to un-

[1] PropBank annotates modals and negation words as modifiers. Since these are not arguments in the common usage of the term, we exclude them from the discussion in this paper.

supervised detection of multiword expressions (MWEs). An important MWE sub-class is that of phrasal verbs, which are also characterized by verb-preposition pairs (Li et al., 2003; Sporleder and Li, 2009) (see also (Boukobza and Rappoport, 2009)). Both tasks aim to determine semantic compositionality, which is a highly challenging task.

Few works addressed unsupervised SRL-related tasks. The setup of (Grenager and Manning, 2006), who presented a Bayesian Network model for argument classification, is perhaps closest to ours. Their work relied on a supervised parser and a rule-based argument identification (both during training and testing). Swier and Stevenson (2004, 2005), while addressing an unsupervised SRL task, greatly differ from us as their algorithm uses the VerbNet (Kipper et al., 2000) verb lexicon, in addition to supervised parses. Finally, Abend et al. (2009) tackled the argument identification task alone and did not perform argument classification of any sort.

PP attachment. PP attachment is the task of determining whether a prepositional phrase which immediately follows a noun phrase attaches to the latter or to the preceding verb. This task's relation to the core-adjunct distinction was addressed in several works. For instance, the results of (Hindle and Rooth, 1993) indicate that their PP attachment system works better for cores than for adjuncts.

Merlo and Esteve Ferrer (2006) suggest a system that jointly tackles the PP attachment and the core-adjunct distinction tasks. Unlike in this work, their classifier requires extensive supervision including WordNet, language-specific features and a supervised parser. Their features are generally motivated by common linguistic considerations. Features found adaptable to a completely unsupervised scenario are used in this work as well.

Syntactic Parsing. The core-adjunct distinction is included in many syntactic annotation schemes. Although the Penn Treebank does not explicitly annotate adjuncts and cores, a few works suggested mapping its annotation (including function tags) to core-adjunct labels. Such a mapping was presented in (Collins, 1999). In his Model 2, Collins modifies his parser to provide a core-adjunct prediction, thereby improving its performance.

The Combinatory Categorial Grammar (CCG) formulation models the core-adjunct distinction explicitly. Therefore, any CCG parser can be used as a core-adjunct classifier (Hockenmaier, 2003).

Subcategorization Acquisition. This task specifies for each predicate the number, type and order of obligatory arguments. Determining the allowable subcategorization frames for a given predicate necessarily involves separating its cores from its allowable adjuncts (which are not framed). Notable works in the field include (Briscoe and Carroll, 1997; Sarkar and Zeman, 2000; Korhonen, 2002). All these works used a parsed corpus in order to collect, for each predicate, a set of hypothesized subcategorization frames, to be filtered by hypothesis testing methods.

This line of work differs from ours in a few aspects. First, all works use manual or supervised syntactic annotations, usually including a POS tagger. Second, the common approach to the task focuses on syntax and tries to identify the entire frame, rather than to tag each argument separately. Finally, most works address the task at the verb type level, trying to detect the allowable frames for each type. Consequently, the common evaluation focuses on the quality of the allowable frames acquired for each verb type, and not on the classification of specific arguments in a given corpus. Such a token level evaluation was conducted in a few works (Briscoe and Carroll, 1997; Sarkar and Zeman, 2000), but often with a small number of verbs or a small number of frames. A discussion of the differences between type and token level evaluation can be found in (Reichart et al., 2010).

The core-adjunct distinction task was tackled in the context of child language acquisition. Villavicencio (2002) developed a classifier based on preposition selection and frequency information for modeling the distinction for locative prepositional phrases. Her approach is not entirely corpus based, as it assumes the input sentences are given in a basic logical form.

The study of prepositions is a vibrant research area in NLP. A special issue of *Computational Linguistics*, which includes an extensive survey of related work, was recently devoted to the field (Baldwin et al., 2009).

3 Algorithm

We are given a (predicate, argument) pair in a test sentence, and we need to determine whether the argument is a core or an adjunct. Test arguments are assumed to be correctly bracketed. We are allowed to utilize a training corpus of raw text.

3.1 Overview

Our algorithm utilizes statistics based on the (predicate, slot, argument head) (PSH) joint distribution (a slot is represented by its preposition). To estimate this joint distribution, PSH samples are extracted from the training corpus using unsupervised POS taggers (Clark, 2003; Abend et al., 2010) and an unsupervised parser (Seginer, 2007). As current performance of unsupervised parsers for long sentences is low, we use only short sentences (up to 10 words, excluding punctuation). The length of test sentences is not bounded. Our results will show that the training data accounts well for the argument realization phenomena in the test set, despite the length bound on its sentences. The sample extraction process is detailed in Section 3.2.

Our approach makes use of both aspects of the distinction – obligatoriness and compositionality. We define three measures, one quantifying the obligatoriness of the slot, another quantifying the selectional preference of the verb to the argument and a third that quantifies the association between the head word and the slot irrespective of the predicate (Section 3.3).

The measures' predictions are expected to coincide in clear cases, but may be less successful in others. Therefore, an ensemble-based method is used to combine the three measures into a single classifier. This results in a high accuracy classifier with relatively low coverage. A self-training step is now performed to increase coverage with only a minor deterioration in accuracy (Section 3.4).

We focus on prepositional arguments. Non-prepositional arguments in English tend to be cores (e.g., in more than 85% of the cases in PB sections 2–21), while prepositional arguments tend to be equally divided between cores and adjuncts. The difficulty of the task thus lies in the classification of prepositional arguments.

3.2 Data Collection

The statistical measures used by our classifier are based on the (predicate, slot, argument head) (PSH) joint distribution. This section details the process of extracting samples from this joint distribution given a raw text corpus.

We start by parsing the corpus using the Seginer parser (Seginer, 2007). This parser is unique in its ability to induce a bracketing (unlabeled parsing) from raw text (without even using POS tags) with strong results. Its high speed (thousands of words per second) allows us to use millions of sentences, a prohibitive number for other parsers.

We continue by tagging the corpus using Clark's unsupervised POS tagger (Clark, 2003) and the unsupervised Prototype Tagger (Abend et al., 2010)[2]. The classes corresponding to prepositions and to verbs are manually selected from the induced clusters[3]. A preposition is defined to be any word which is the first word of an argument and belongs to a prepositions cluster. A verb is any word belonging to a verb cluster. This manual selection requires only a minute, since the number of classes is very small (34 in our experiments). In addition, knowing what is considered a preposition is part of the task definition itself.

Argument identification is hard even for supervised models and is considerably more so for unsupervised ones (Abend et al., 2009). We therefore confine ourselves to sentences of length not greater than 10 (excluding punctuation) which contain a single verb. A sequence of words will be marked as an argument of the verb if it is a constituent that does not contain the verb (according to the unsupervised parse tree), whose parent is an ancestor of the verb. This follows the pruning heuristic of (Xue and Palmer, 2004) often used by SRL algorithms.

The corpus is now tagged using an unsupervised POS tagger. Since the sentences in question are short, we consider every word which does not belong to a closed class cluster as a head word (an argument can have several head words). A closed class is a class of function words with relatively few word types, each of which is very frequent. Typical examples include determiners, prepositions and conjunctions. A class which is not closed is open. In this paper, we define closed classes to be clusters in which the ratio between the number of word tokens and the number of word types ex-

[2]Clark's tagger was replaced by the Prototype Tagger where the latter gave a significant improvement. See Section 4.

[3]We also explore a scenario in which they are identified by a supervised tagger. See Section 4.

ceeds a threshold T[4].

Using these annotation layers, we traverse the corpus and extract every (predicate, slot, argument head) triplet. In case an argument has several head words, each of them is considered as an independent sample. We denote the number of times that a triplet occurred in the training corpus by $N(p, s, h)$.

3.3 Collocation Measures

In this section we present the three types of measures used by the algorithm and the rationale behind each of them. These measures are all based on the PSH joint distribution.

Given a (predicate, prepositional argument) pair from the test set, we first tag and parse the argument using the unsupervised tools above[5]. Each word in the argument is now represented by its word form (without lemmatization), its unsupervised POS tag and its depth in the parse tree of the argument. The last two will be used to determine which are the head words of the argument (see below). The head words themselves, once chosen, are represented by the lemma. We now compute the following measures.

Selectional Preference (SP). Since the semantics of cores is more predicate dependent than the semantics of adjuncts, we expect arguments for which the predicate has a strong preference (in a specific slot) to be cores.

Selectional preference induction is a well-established task in NLP. It aims to quantify the likelihood that a certain argument appears in a certain slot of a predicate. Several methods have been suggested (Resnik, 1996; Li and Abe, 1998; Schulte im Walde et al., 2008).

We use the paradigm of (Erk, 2007). For a given predicate slot pair (p, s), we define its preference to the argument head h to be:

$$SP(p, s, h) = \sum_{h' \in Heads} Pr(h'|p, s) \cdot sim(h, h')$$

$$Pr(h|p, s) = \frac{N(p, s, h)}{\Sigma_{h'} N(p, s, h')}$$

$sim(h, h')$ is a similarity measure between argument heads. $Heads$ is the set of all head words. This is a natural extension of the naive (and sparse) maximum likelihood estimator $Pr(h|p, s)$, which is obtained by taking $sim(h, h')$ to be 1 if $h = h'$ and 0 otherwise.

The similarity measure we use is based on the slot distributions of the arguments. That is, two arguments are considered similar if they tend to appear in the same slots. Each head word h is assigned a vector where each coordinate corresponds to a slot s. The value of the coordinate is the number of times h appeared in s, i.e. $\Sigma_{p'} N(p', s, h)$ (p' is summed over all predicates). The similarity measure between two head words is then defined as the cosine measure of their vectors.

Since arguments in the test set can be quite long, not every open class word in the argument is taken to be a head word. Instead, only those appearing in the top level (depth = 1) of the argument under its unsupervised parse tree are taken. In case there are no such open class words, we take those appearing in depth 2. The selectional preference of the whole argument is then defined to be the arithmetic mean of this measure over all of its head words. If the argument has no head words under this definition or if none of the head words appeared in the training corpus, the selectional preference is undefined.

Predicate-Slot Collocation. Since cores are obligatory, when a predicate persistently appears with an argument in a certain slot, the arguments in this slot tends to be cores. This notion can be captured by the $(predicate, slot)$ joint distribution. We use the Pointwise Mutual Information measure (PMI) to capture the slot and the predicate's collocation tendency. Let p be a predicate and s a slot, then:

$$PS(p, s) = PMI(p, s) = log \frac{Pr(p, s)}{Pr(s) \cdot Pr(p)} =$$

$$= log \frac{N(p, s) \Sigma_{p', s'} N(p', s')}{\Sigma_{s'} N(p, s') \Sigma_{p'} N(p', s)}$$

Since there is only a meager number of possible slots (that is, of prepositions), estimating the $(predicate, slot)$ distribution can be made by the maximum likelihood estimator with manageable sparsity.

In order not to bias the counts towards predicates which tend to take more arguments, we define here $N(p, s)$ to be the number of times the (p, s) pair occurred in the training corpus, irrespective of the number of head words the argument had (and not e.g., $\Sigma_h N(p, s, h)$). Argu-

[4] We use sections 2–21 of the PTB WSJ for these counts, containing 0.95M words. Our T was set to 50.

[5] Note that while current unsupervised parsers have low performance on long sentences, arguments, even in long sentences, are usually still short enough for them to operate well. Their average length in the test set is 5.1 words.

ments with no prepositions are included in these counts as well (with $s = NULL$), so not to bias against predicates which tend to have less non-prepositional arguments.

Argument-Slot Collocation. Adjuncts tend to belong to one of a few specific semantic domains (see Section 2). Therefore, if an argument tends to appear in a certain slot in many of its instances, it is an indication that this argument tends to have a consistent semantic flavor in most of its instances. In this case, the argument and the preposition can be viewed as forming a unit on their own, independent of the predicate with which they appear. We therefore expect such arguments to be adjuncts.

We formalize this notion using the following measure. Let p, s, h be a predicate, a slot and a head word respectively. We then use[6]:

$$AS(s,h) = 1 - Pr(s|h) = 1 - \frac{\Sigma_{p'} N(p', s, h)}{\Sigma_{p',s'} N(p', s', h)}$$

We select the head words of the argument as we did with the selectional preference measure. Again, the AS of the whole argument is defined to be the arithmetic mean of the measure over all of its head words.

Thresholding. In order to turn these measures into classifiers, we set a threshold below which arguments are marked as adjuncts and above which as cores. In order to avoid tuning a parameter for each of the measures, we set the threshold as the median value of this measure in the test set. That is, we find the threshold which tags half of the arguments as cores and half as adjuncts. This relies on the prior knowledge that prepositional arguments are roughly equally divided between cores and adjuncts[7].

3.4 Combination Model

The algorithm proceeds to integrate the predictions of the weak classifiers into a single classifier. We use an ensemble method (Breiman, 1996). Each of the classifiers may either classify an argument as an adjunct, classify it as a core, or abstain. In order to obtain a high accuracy classifier, to be used for self-training below, the ensemble classifier only tags arguments for which none of the classifiers abstained, i.e., when sufficient information was available to make all three predictions. The prediction is determined by the majority vote.

The ensemble classifier has high precision but low coverage. In order to increase its coverage, a self-training step is performed. We observe that a predicate and a slot generally determine whether the argument is a core or an adjunct. For instance, in our development data, a classifier which assigns all arguments that share a predicate and a slot their most common label, yields 94.3% accuracy on the pairs appearing at least 5 times. This property of the core-adjunct distinction greatly simplifies the task for supervised algorithms (see Section 2).

We therefore apply the following procedure: (1) tag the training data with the ensemble classifier; (2) for each test sample x, if more than a ratio of α of the training samples sharing the same predicate and slot with x are labeled as cores, tag x as core. Otherwise, tag x as adjunct.

Test samples which do not share a predicate and a slot with any training sample are considered out of coverage. The parameter α is chosen so half of the arguments are tagged as cores and half as adjuncts. In our experiments α was about 0.25.

4 Experimental Setup

Experiments were conducted in two scenarios. In the '*SID*' (supervised identification of prepositions and verbs) scenario, a gold standard list of prepositions was provided. The list was generated by taking every word tagged by the preposition tag ('*IN*') in at least one of its instances under the gold standard annotation of the WSJ sections 2–21. Verbs were identified using MXPOST (Ratnaparkhi, 1996). Words tagged with any of the verb tags, except of the auxiliary verbs ('have', 'be' and 'do') were considered predicates. This scenario decouples the accuracy of the algorithm from the quality of the unsupervised POS tagging.

In the '*Fully Unsupervised*' scenario, prepositions and verbs were identified using Clark's tagger (Clark, 2003). It was asked to produce a tagging into 34 classes. The classes corresponding to prepositions and to verbs were manually identified. Prepositions in the test set were detected with 84.2% precision and 91.6% recall.

The prediction of whether a word belongs to an open class or a closed was based on the output of the Prototype tagger (Abend et al., 2010). The Prototype tagger provided significantly more ac-

[6] The conditional probability is subtracted from 1 so that higher values correspond to cores, as with the other measures.

[7] In case the test data is small, we can use the median value on the training data instead.

curate predictions in this context than Clark's.

The 39832 sentences of PropBank's sections 2–21 were used as a test set without bounding their lengths[8]. Cores were defined to be any argument bearing the labels 'A0' – 'A5', 'C-A0' – 'C-A5' or 'R-A0' – 'R-A5'. Adjuncts were defined to be arguments bearing the labels 'AM', 'C-AM' or 'R-AM'. Modals ('AM-MOD') and negation modifiers ('AM-NEG') were omitted since they do not represent adjuncts.

The test set includes 213473 arguments, 45939 (21.5%) are prepositional. Of the latter, 22442 (48.9%) are cores and 23497 (51.1%) are adjuncts. The non-prepositional arguments include 145767 (87%) cores and 21767 (13%) adjuncts. The average number of words per argument is 5.1.

The NANC (Graff, 1995) corpus was used as a training set. Only sentences of length not greater than 10 excluding punctuation were used (see Section 3.2), totaling 4955181 sentences. 7673878 (5635810) arguments were identified in the '*SID*' ('*Fully Unsupervised*') scenario. The average number of words per argument is 1.6 (1.7).

Since this is the first work to tackle this task using neither manual nor supervised syntactic annotation, there is no previous work to compare to. However, we do compare against a non-trivial baseline, which closely follows the rationale of cores as obligatory arguments.

Our *Window Baseline* tags a corpus using MXPOST and computes, for each predicate and preposition, the ratio between the number of times that the preposition appeared in a window of W words after the verb and the total number of times that the verb appeared. If this number exceeds a certain threshold β, all arguments having that predicate and preposition are tagged as cores. Otherwise, they are tagged as adjuncts. We used 18.7M sentences from NANC of unbounded length for this baseline. W and β were fine-tuned against the test set[9].

We also report results for partial versions of the algorithm, starting with the three measures used (selectional preference, predicate-slot collocation and argument-slot collocation). Results for the ensemble classifier (prior to the bootstrapping stage) are presented in two variants: one in which the ensemble is used to tag arguments for which all three measures give a prediction (the '*Ensemble(Intersection)*' classifier) and one in which the ensemble tags all arguments for which at least one classifier gives a prediction (the '*Ensemble(Union)*' classifier). For the latter, a tie is broken in favor of the core label. The '*Ensemble(Union)*' classifier is not a part of our model and is evaluated only as a reference.

In order to provide a broader perspective on the task, we compare the measures in the basis of our algorithm to simplified or alternative measures. We experiment with the following measures:

1. *Simple SP* – a selectional preference measure defined to be $Pr(head|slot, predicate)$.

2. *Vast Corpus SP* – similar to '*Simple SP*' but with a much larger corpus. It uses roughly 100M arguments which were extracted from the web-crawling based corpus of (Gabrilovich and Markovitch, 2005) and the British National Corpus (Burnard, 2000).

3. *Thesaurus SP* – a selectional preference measure which follows the paradigm of (Erk, 2007) (Section 3.3) and defines the similarity between two heads to be the Jaccard affinity between their two entries in Lin's automatically compiled thesaurus (Lin, 1998)[10].

4. *Pr(slot|predicate)* – an alternative to the used predicate-slot collocation measure.

5. *PMI(slot, head)* – an alternative to the used argument-slot collocation measure.

6. *Head Dependence* – the entropy of the predicate distribution given the slot and the head (following (Merlo and Esteve Ferrer, 2006)):

$$HD(s,h) = -\Sigma_p Pr(p|s,h) \cdot log(Pr(p|s,h))$$

Low entropy implies a core.

For each of the scenarios and the algorithms, we report accuracy, coverage and effective accuracy. Effective accuracy is defined to be the accuracy obtained when all out of coverage arguments are tagged as adjuncts. This procedure always yields a classifier with 100% coverage and therefore provides an even ground for comparing the algorithms' performance.

We see accuracy as important on its own right since increasing coverage is often straightforward given easily obtainable larger training corpora.

[8]The first 15K arguments were used for the algorithm's development and therefore excluded from the evaluation.

[9]Their optimal value was found to be W=2, β=0.03. The low optimal value of β is an indication of the noisiness of this technique.

[10]Since we aim for a minimally supervised scenario, we used the proximity-based version of his thesaurus which does not require parsing as pre-processing. http://webdocs.cs.ualberta.ca/∼lindek/Downloads/sims.lsp.gz

		Collocation Measures				Ensemble + Cov.	
		Sel. Preference	Pred-Slot	Arg-Slot	Ensemble(I)	Ensemble(U)	E(I) + ST
SID Scenario	Accuracy	65.6	64.5	72.4	**74.1**	68.7	70.6
	Coverage	35.6	77.8	44.7	33.2	**88.1**	74.2
	Eff. Acc.	56.7	64.8	58.8	58.8	67.8	**68.4**
Fully Unsupervised Scenario	Accuracy	62.6	61.1	69.4	**70.6**	64.8	68.8
	Coverage	24.8	59.0	38.7	22.8	**74.2**	56.9
	Eff. Acc.	52.6	57.5	55.8	53.8	61.0	**61.4**

Table 1: Results for the various models. Accuracy, coverage and effective accuracy are presented in percents. Effective accuracy is defined to be the accuracy resulting from labeling each out of coverage argument with an adjunct label. The rows represent the following models (left to right): selectional preference, predicate-slot collocation, argument-slot collocation, '*Ensemble(Intersection)*', '*Ensemble(Union)*' and the '*Ensemble(Intersection)*' followed by self-training (see Section 3.4). '*Ensemble(Intersection)*' obtains the highest accuracy. The ensemble + self-training obtains the highest effective accuracy.

	Selectional Preference Measures				Pred-Slot Measures			Arg-Slot Measures		
	SP*	S. SP	V.C. SP	Lin SP	PS*	Pr(s|p)	Window	AS*	PMI(s, h)	HD
Acc.	**65.6**	41.6	44.8	49.9	**64.5**	58.9	64.1	**72.4**	67.5	67.4
Cov.	35.6	36.9	**45.3**	36.7	77.8	77.8	**92.6**	44.7	44.7	44.7
Eff. Acc.	**56.7**	48.2	47.7	51.3	64.8	60.5	**65.0**	**58.8**	56.6	56.6

Table 2: Comparison of the measures used by our model to alternative measures in the '*SID*' scenario. Results are in percents. The sections of the table are (from left to right): selectional preference measures, predicate-slot measures, argument-slot measures and head dependence. The measures are (left to right): SP*, Simple SP, Vast Corpus SP, Lin SP, PS*, Pr(slot|predicate), Window Baseline, AS*, PMI(slot, head) and Head Dependence. The measures marked with $*$ are the ones used by our model. See Section 4.

Another reason is that a high accuracy classifier may provide training data to be used by subsequent supervised algorithms.

For completeness, we also provide results for the entire set of arguments. The great majority of non-prepositional arguments are cores (87% in the test set). We therefore tag all non-prepositional as cores and tag prepositional arguments using our model. In order to minimize supervision, we distinguish between the prepositional and the non-prepositional arguments using Clark's tagger.

Finally, we experiment on a scenario where even argument identification on the test set is not provided, but performed by the algorithm of (Abend et al., 2009), which uses neither syntactic nor SRL annotation but does utilize a supervised POS tagger. We therefore run it in the '*SID*' scenario. We apply it to the sentences of length at most 10 contained in sections 2–21 of PB (11586 arguments in 6007 sentences). Non-prepositional arguments are invariably tagged as cores and out of coverage prepositional arguments as adjuncts.

We report labeled and unlabeled recall, precision and F-scores for this experiment. An unlabeled match is defined to be an argument that agrees in its boundaries with a gold standard argument and a labeled match requires in addition that the arguments agree in their core/adjunct label. We also report labeling accuracy which is the ratio between the number of labeled matches and the number of unlabeled matches[11].

5 Results

Table 1 presents the results of our main experiments. In both scenarios, the most accurate of the three basic classifiers was the argument-slot collocation classifier. This is an indication that the collocation between the argument and the preposition is more indicative of the core/adjunct label than the obligatoriness of the slot (as expressed by the predicate-slot collocation).

Indeed, we can find examples where adjuncts, although optional, appear very often with a certain verb. An example is 'meet', which often takes a temporal adjunct, as in 'Let's meet [in July]'. This is a semantic property of 'meet', whose syntactic expression is not obligatory.

All measures suffered from a comparable deterioration of accuracy when moving from the '*SID*' to the '*Fully Unsupervised*' scenario. The deterioration in coverage, however, was considerably lower for the argument-slot collocation.

The '*Ensemble(Intersection)*' model in both cases is more accurate than each of the basic classifiers alone. This is to be expected as it combines the predictions of all three. The self-training step significantly increases the ensemble model's cov-

[11] Note that the reported unlabeled scores are slightly lower than those reported in the 2009 paper, due to the exclusion of the modals and negation modifiers.

	Precision	Recall	F-score	lAcc.
Unlabeled	50.7	66.3	57.5	–
Labeled	42.4	55.4	48.0	83.6

Table 3: Unlabeled and labeled scores for the experiments using the unsupervised argument identification system of (Abend et al., 2009). Precision, recall, F-score and labeling accuracy are given in percents.

erage (with some loss in accuracy), thus obtaining the highest effective accuracy. It is also more accurate than the simpler classifier '*Ensemble(Union)*' (although the latter's coverage is higher).

Table 2 presents results for the comparison to simpler or alternative measures. Results indicate that the three measures used by our algorithm (leftmost column in each section) obtain superior results. The only case in which performance is comparable is the window baseline compared to the Pred-Slot measure. However, the baseline's score was obtained by using a much larger corpus and a careful hand-tuning of the parameters[12].

The poor performance of *Simple SP* can be ascribed to sparsity. This is demonstrated by the median value of 0, which this measure obtained on the test set. Accuracy is only somewhat better with a much larger corpus (*Vast Corpus SP*). The *Thesaurus SP* most probably failed due to insufficient coverage, despite its applicability in a similar supervised task (Zapirain et al., 2009).

The Head Dependence measure achieves a relatively high accuracy of 67.4%. We therefore attempted to incorporate it into our model, but failed to achieve a significant improvement to the overall result. We expect a further study of the relations between the measures will suggest better ways of combining their predictions.

The obtained effective accuracy for the entire set of arguments, where the prepositional arguments are automatically identified, was 81.6%.

Table 3 presents results of our experiments with the unsupervised argument identification model of (Abend et al., 2009). The unlabeled scores reflect performance on argument identification alone, while the labeled scores reflect the joint performance of both the 2009 and our algorithms. These results, albeit low, are potentially beneficial for unsupervised subcategorization acquisition. The accuracy of our model on the entire set (prepositional argument subset) of correctly identified arguments was 83.6% (71.7%). This is somewhat higher than the score on the entire test set ('*SID*' scenario), which was 83.0% (68.4%), probably due to the bounded length of the test sentences in this case.

[12]We tried about 150 parameter pairs for the baseline. The average of the five best effective accuracies was 64.3%.

6 Conclusion

We presented a fully unsupervised algorithm for the classification of arguments into cores and adjuncts. Since most non-prepositional arguments are cores, we focused on prepositional arguments, which are roughly equally divided between cores and adjuncts. The algorithm computes three statistical measures and utilizes ensemble-based and self-training methods to combine their predictions.

The algorithm applies state-of-the-art unsupervised parser and POS tagger to collect statistics from a large raw text corpus. It obtains an accuracy of roughly 70%. We also show that (somewhat surprisingly) an argument-slot collocation measure gives more accurate predictions than a predicate-slot collocation measure on this task. We speculate the reason is that the head word disambiguates the preposition and that this disambiguation generally determines whether a prepositional argument is a core or an adjunct (somewhat independently of the predicate). This calls for a future study into the semantics of prepositions and their relation to the core-adjunct distinction. In this context two recent projects, *The Preposition Project* (Litkowski and Hargraves, 2005) and *PrepNet* (Saint-Dizier, 2006), which attempt to characterize and categorize the complex syntactic and semantic behavior of prepositions, may be of relevance.

It is our hope that this work will provide a better understanding of core-adjunct phenomena. Current supervised SRL models tend to perform worse on adjuncts than on cores (Pradhan et al., 2008; Toutanova et al., 2008). We believe a better understanding of the differences between cores and adjuncts may contribute to the development of better SRL techniques, in both its supervised and unsupervised variants.

References

Omri Abend, Roi Reichart and Ari Rappoport, 2009. *Unsupervised Argument Identification for Semantic Role Labeling.* ACL '09.

Omri Abend, Roi Reichart and Ari Rappoport, 2010. *Improved Unsupervised POS Induction through Prototype Discovery.* ACL '10.

Collin F. Baker, Charles J. Fillmore and John B. Lowe, 1998. *The Berkeley FrameNet Project.* ACL-COLING '98.

Timothy Baldwin, Valia Kordoni and Aline Villavicencio, 2009. *Prepositions in Applications: A Survey and Introduction to the Special Issue.* Computational Linguistics, 35(2):119–147.

Ram Boukobza and Ari Rappoport, 2009. *Multi-Word Expression Identification Using Sentence Surface Features.* EMNLP '09.

Leo Breiman, 1996. *Bagging Predictors.* Machine Learning, 24(2):123–140.

Ted Briscoe and John Carroll, 1997. *Automatic Extraction of Subcategorization from Corpora.* Applied NLP '97.

Lou Burnard, 2000. *User Reference Guide for the British National Corpus.* Technical report, Oxford University.

Xavier Carreras and Lluìs Màrquez, 2005. *Introduction to the CoNLL–2005 Shared Task: Semantic Role Labeling.* CoNLL '05.

Alexander Clark, 2003. *Combining Distributional and Morphological Information for Part of Speech Induction.* EACL '03.

Michael Collins, 1999. *Head-driven statistical models for natural language parsing.* Ph.D. thesis, University of Pennsylvania.

David Dowty, 2000. *The Dual Analysis of Adjuncts and Complements in Categorial Grammar.* Modifying Adjuncts, ed. Lang, Maienborn and Fabricius–Hansen, de Gruyter, 2003.

Katrin Erk, 2007. *A Simple, Similarity-based Model for Selectional Preferences.* ACL '07.

Evgeniy Gabrilovich and Shaul Markovitch, 2005. *Feature Generation for Text Categorization using World Knowledge.* IJCAI '05.

David Graff, 1995. *North American News Text Corpus.* Linguistic Data Consortium. LDC95T21.

Trond Grenager and Christopher D. Manning, 2006. *Unsupervised Discovery of a Statistical Verb Lexicon.* EMNLP '06.

Donald Hindle and Mats Rooth, 1993. *Structural Ambiguity and Lexical Relations.* Computational Linguistics, 19(1):103–120.

Julia Hockenmaier, 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar.* Ph.D. thesis, University of Edinburgh.

Karin Kipper, Hoa Trang Dang and Martha Palmer, 2000. *Class-Based Construction of a Verb Lexicon.* AAAI '00.

Anna Korhonen, 2002. *Subcategorization Acquisition.* Ph.D. thesis, University of Cambridge.

Hang Li and Naoki Abe, 1998. *Generalizing Case Frames using a Thesaurus and the MDL Principle.* Computational Linguistics, 24(2):217–244.

Wei Li, Xiuhong Zhang, Cheng Niu, Yuankai Jiang and Rohini Srihari, 2003. *An Expert Lexicon Approach to Identifying English Phrasal Verbs.* ACL '03.

Dekang Lin, 1998. *Automatic Retrieval and Clustering of Similar Words.* COLING–ACL '98.

Ken Litkowski and Orin Hargraves, 2005. *The Preposition Project.* ACL-SIGSEM Workshop on "The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications".

Diana McCarthy, 2001. *Lexical Acquisition at the Syntax-Semantics Interface: Diathesis Alternations, Subcategorization Frames and Selectional Preferences.* Ph.D. thesis, University of Sussex.

Paula Merlo and Eva Esteve Ferrer, 2006. *The Notion of Argument in Prepositional Phrase Attachment.* Computational Linguistics, 32(3):341–377.

Martha Palmer, Daniel Gildea and Paul Kingsbury, 2005. *The Proposition Bank: A Corpus Annotated with Semantic Roles.* Computational Linguistics, 31(1):71–106.

Sameer Pradhan, Wayne Ward and James H. Martin, 2008. *Towards Robust Semantic Role Labeling.* Computational Linguistics, 34(2):289–310.

Vasin Punyakanok, Dan Roth and Wen-tau Yih, 2008. *The Importance of Syntactic Parsing and Inference in Semantic Role Labeling.* Computational Linguistics, 34(2):257–287.

Adwait Ratnaparkhi, 1996. *Maximum Entropy Part-Of-Speech Tagger.* EMNLP '96.

Roi Reichart, Omri Abend and Ari Rappoport, 2010. *Type Level Clustering Evaluation: New Measures and a POS Induction Case Study.* CoNLL '10.

Philip Resnik, 1996. *Selectional constraints: An information-theoretic model and its computational realization.* Cognition, 61:127–159.

Patrick Saint-Dizier, 2006. *PrepNet: A Multilingual Lexical Description of Prepositions.* LREC '06.

Anoop Sarkar and Daniel Zeman, 2000. *Automatic Extraction of Subcategorization Frames for Czech.* COLING '00.

Sabine Schulte im Walde, Christian Hying, Christian Scheible and Helmut Schmid, 2008. *Combining EM Training and the MDL Principle for an Automatic Verb Classification Incorporating Selectional Preferences.* ACL '08.

Yoav Seginer, 2007. *Fast Unsupervised Incremental Parsing.* ACL '07.

Caroline Sporleder and Linlin Li, 2009. *Unsupervised Recognition of Literal and Non-Literal Use of Idiomatic Expressions.* EACL '09.

Robert S. Swier and Suzanne Stevenson, 2004. *Unsupervised Semantic Role Labeling.* EMNLP '04.

Robert S. Swier and Suzanne Stevenson, 2005. *Exploiting a Verb Lexicon in Automatic Semantic Role Labelling.* EMNLP '05.

Kristina Toutanova, Aria Haghighi and Christopher D. Manning, 2008. *A Global Joint Model for Semantic Role Labeling.* Computational Linguistics, 34(2):161–191.

Aline Villavicencio, 2002. *Learning to Distinguish PP Arguments from Adjuncts.* CoNLL '02.

Dave Willis, 2004. *Collins Cobuild Intermedia English Grammar, Second Edition*. HarperCollins Publishers.

Nianwen Xue and Martha Palmer, 2004. *Calibrating Features for Semantic Role Labeling.* EMNLP '04.

Beñat Zapirain, Eneko Agirre and Lluís Màrquez, 2009. *Generalizing over Lexical Features: Selectional Preferences for Semantic Role Classification.* ACL '09, short paper.

Towards Open-Domain Semantic Role Labeling

Danilo Croce, Cristina Giannone, Paolo Annesi, Roberto Basili
{croce,giannone,annesi,basili}@info.uniroma2.it
Department of Computer Science, Systems and Production
University of Roma, *Tor Vergata*

Abstract

Current Semantic Role Labeling technologies are based on inductive algorithms trained over large scale repositories of annotated examples. Frame-based systems currently make use of the FrameNet database but fail to show suitable generalization capabilities in out-of-domain scenarios. In this paper, a state-of-art system for frame-based SRL is extended through the encapsulation of a distributional model of semantic similarity. The resulting argument classification model promotes a simpler feature space that limits the potential overfitting effects. The large scale empirical study here discussed confirms that state-of-art accuracy can be obtained for out-of-domain evaluations.

1 Introduction

The availability of large scale semantic lexicons, such as FrameNet (Baker et al., 1998), allowed the adoption of a wide family of learning paradigms in the automation of semantic parsing. Building upon the so called *frame* semantic model (Fillmore, 1985), the Berkeley FrameNet project has developed a semantic lexicon for the core vocabulary of English, since 1997. A frame is evoked in texts through the occurrence of its *lexical units* (LU), i.e. predicate words such verbs, nouns, or adjectives, and specifies the participants and properties of the situation it describes, the so called *frame elements* (FEs).

Semantic Role Labeling (SRL) is the task of automatic recognition of individual predicates together with their major roles (e.g. frame elements) as they are grammatically realized in input sentences. It has been a popular task since the availability of the PropBank and FrameNet annotated corpora (Palmer et al., 2005), the seminal work of (Gildea and Jurafsky, 2002) and the successful CoNLL evaluation campaigns (Carreras and Màrquez, 2005). Statistical machine learning methods, ranging from joint probabilistic models to support vector machines, have been successfully adopted to provide very accurate semantic labeling, e.g. (Carreras and Màrquez, 2005).

SRL based on FrameNet is thus not a novel task, although very few systems are known capable of completing a general frame-based annotation process over raw texts, noticeable exceptions being discussed for example in (Erk and Pado, 2006), (Johansson and Nugues, 2008b) and (Coppola et al., 2009). Some critical limitations have been outlined in literature, some of them independent from the underlying semantic paradigm.

Parsing Accuracy. Most of the employed learning algorithms are based on complex sets of syntagmatic features, as deeply investigated in (Johansson and Nugues, 2008b). The resulting recognition *is thus highly dependent on the accuracy of the underlying parser*, whereas wrong structures returned by the parser usually imply large misclassification errors.

Annotation costs. Statistical learning approaches applied to SRL are *very demanding with respect to the amount and quality of the training material*. The complex SRL architectures proposed (usually combining local and global, i.e. joint, models of argument classification, e.g. (Toutanova et al., 2008)) require a large number of annotated examples. The amount and quality of the training data required to reach a significant accuracy is a serious limitation to the exploitation of SRL in many NLP applications.

Limited Linguistic Generalization. Several studies showed that even when large training sets exist *the corresponding learning exhibits poor generalization power*. Most of the CoNLL 2005 systems show a significant performance drop when the tested corpus, i.e. Brown, differs from

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 237–246,
Uppsala, Sweden, 11-16 July 2010.

the training one (i.e. Wall Street Journal), e.g. (Toutanova et al., 2008). More recently, the state-of-art frame-based semantic role labeling system discussed in (Johansson and Nugues, 2008b) reports a 19% drop in accuracy for the argument classification task when a different test domain is targeted (i.e. NTI corpus). Out-of-domain tests seem to suggest the models trained on BNC do not generalize well to novel grammatical and lexical phenomena. As also suggested in (Pradhan et al., 2008), the major drawback is the poor generalization power affecting lexical features. Notice how this is also a general problem of statistical learning processes, as large fine grain feature sets are more exposed to the risks of overfitting.

The above problems are particularly critical for frame-based shallow semantic parsing where, as opposed to more syntactic-oriented semantic labeling schemes (as Propbank (Palmer et al., 2005)), *a significant mismatch exists between the semantic descriptors and the underlying syntactic annotation level*. In (Johansson and Nugues, 2008b) an upper bound of about 83.9% for the accuracy of the argument identification task is reported, it is due to the complexity in projecting frame element boundaries out from the dependency graph: more than 16% of the roles in the annotated material lack of a clear grammatical status.

The limited level of linguistic generalization outlined above is still an open research problem. Existing solutions have been proposed in literature along different lines. Learning from richer linguistic descriptions of more complex structures is proposed in (Toutanova et al., 2008). Limiting the cost required for developing large domain-specific training data sets has been also studied, e.g., (Fürstenau and Lapata, 2009). Finally, the application of semi-supervised learning is attempted to increase the lexical expressiveness of the model, e.g. (Goldberg and Elhadad, 2009).

In this paper, this last direction is pursued. A semi-supervised statistical model exploiting useful lexical information from unlabeled corpora is proposed. The model adopts a simple feature space by relying on a limited set of grammatical properties, thus reducing its learning capacity. Moreover, it generalizes lexical information about the annotated examples by applying a geometrical model, in a Latent Semantic Analysis style, inspired by a distributional paradigm (Pado and Lapata, 2007). As we will see, the accuracy reachable through a restricted feature space is still quite close to the state-of-art, but interestingly the performance drops in out-of-domain tests are avoided.

In the following, after discussing existing approaches to SRL (Section 2), a distributional approach is defined in Section 3. Section 3.2 discusses the proposed HMM-based treatment of joint inferences in argument classification. The large scale experiments described in Section 4 will allow to draw the conclusions of Section 5.

2 Related Work

State-of-art approaches to frame-based SRL are based on Support Vector Machines, trained over linear models of syntactic features, e.g. (Johansson and Nugues, 2008b), or tree-kernels, e.g. (Coppola et al., 2009). SRL proceeds through two main steps: the localization of arguments in a sentence, called *boundary detection (BD)*, and the assignment of the proper role to the detected constituents, that is the *argument classification, (AC)* step. In (Toutanova et al., 2008) a SRL model over Propbank that effectively exploits the semantic argument frame as a joint structure, is presented. It incorporates strong dependencies within a comprehensive statistical joint model with a rich set of features over multiple argument phrases. This approach effectively introduces a new step in SRL, also called *Joint Re-ranking, (RR)*, e.g. (Toutanova et al., 2008) or (Moschitti et al., 2008). First local models are applied to produce role labels over individual arguments, then the joint model is used to decide the entire argument sequence among the set of the n-best competing solutions. While these approaches increase the expressive power of the models to capture more general linguistic properties, they rely on complex feature sets, are more demanding about the amount of training information and increase the overall exposure to overfitting effects.

In (Johansson and Nugues, 2008b) the impact of different grammatical representations on the task of frame-based shallow semantic parsing is studied and the poor lexical generalization problem is outlined. An argument classification accuracy of 89.9% over the FrameNet (i.e. BNC) dataset is shown to decrease to 71.1% when a different test domain is evaluated (i.e. the Nuclear Threat Initiative corpus). The argument classification

component is thus shown to be heavily domain-dependent whereas the inclusion of grammatical function features is just able to mitigate this sensitivity. In line with (Pradhan et al., 2008), it is suggested that lexical features are domain specific and their suitable generalization is not achieved.

The lack of suitable lexical information is also discussed in (Fürstenau and Lapata, 2009) through an approach aiming to support the creation of novel annotated resources. Accordingly a semi-supervised approach for reducing the costs of the manual annotation effort is proposed. Through a graph alignment algorithm triggered by annotated resources, the method acquires training instances from an unlabeled corpus also for verbs not listed as existing FrameNet predicates.

2.1 The role of Lexical Semantic Information

It is widely accepted that lexical information (as features directly derived from word forms) is crucial for training accurate systems in a number of NLP tasks. Indeed, all the best systems in the CoNLL shared task competitions (e.g. Chunking (Tjong Kim Sang and Buchholz, 2000)) make extensive use of lexical information. Also lexical features are beneficial in SRL usually either for systems on Propbank as well as for FrameNet-based annotation.

In (Goldberg and Elhadad, 2009), a different strategy to incorporate lexical features into classification models is proposed. A more expressive training algorithm (i.e. anchored SVM) coupled with an aggressive feature pruning strategy is shown to achieve high accuracy over a chunking and named entity recognition task. The suggested perspective here is that effective semantic knowledge can be collected from sources external to the annotated corpora (very large unannotated corpora or on manually constructed lexical resources) rather than learned from the raw lexical counts of the annotated corpus. Notice how this is also the strategy pursued in recent work on deep learning approaches to NLP tasks. In (Collobert and Weston, 2008) a unified architecture for Natural Language Processing that learns features relevant to the tasks at hand given very limited prior knowledge is presented. It embodies the idea that a multitask learning architecture coupled with semi-supervised learning can be effectively applied even to complex linguistic tasks such as SRL. In particular, (Collobert and Weston, 2008) proposes an embedding of lexical information using Wikipedia as source, and exploiting the resulting language model within the multitask learning process. The idea of (Collobert and Weston, 2008) to obtain an embedding of lexical information by acquiring a language model from unlabeled data is an interesting approach to the problem of performance degradation in out-of-domain tests, as already pursued by (Deschacht and Moens, 2009). The extensive use of unlabeled texts allows to achieve a significant level of lexical generalization that seems better capitalize the smaller annotated data sets.

3 A Distributional Model for Argument Classification

High quality lexical information is crucial for robust open-domain SRL, as semantic generalization highly depends on lexical information. For example, the following two sentences evoke the STATEMENT frame, through the LUs *say* and *state*, where the FEs, SPEAKER and MEDIUM, are shown.

[*President Kennedy*] SPEAKER said *to an astronaut, "Man is still the most extraordinary computer of all."* (1)

[*The report*] MEDIUM stated, *that some problems needed to be solved.* (2)

In sentence (1), for example, *President Kennedy* is the grammatical subject of the verb *say* and this justifies its role of SPEAKER. However, syntax does not entirely characterize argument semantics. In (1) and (2), the same syntactic relation is observed. It is the semantics of the grammatical heads, i.e. *report* and *Kennedy*, the main responsible for the difference between the two resulting proto-agentive roles, SPEAKER and MEDIUM.

In this work we explore two different aspects. First, we propose a model that does not depend on complex syntactic information in order to minimize the risk of overfitting. Second, we improve the lexical semantic information available to the learning algorithm. The proposed "minimalistic" approach will consider only two independent features:

- the *semantic head* (h) of a role, as it can be observed in the grammatical structure. In sentence (2), for example, the MEDIUM FE is realized as the logical subject, whose head is *report*.

- the *dependency relation* (r) connecting the semantic head to the predicate words. In (2), the semantic head *report* is connected to the LU *stated* through the subject (SBJ) relation.

In the rest of the section the distributional model for the argument classification step is presented. A lexicalized model for individual semantic roles is first defined in order to achieve robust semantic classification local to each argument. Then a Hidden Markov Model is introduced in order to exploit the local probability estimators, sensitive to lexical similarity, as well as the global information on the entire argument sequence.

3.1 Distributional Local Models

As the classification of semantic roles is strictly related to the lexical meaning of argument heads, we adopt a distributional perspective, where the meaning is described by the set of textual contexts in which words appear. In distributional models, words are thus represented through vectors built over these observable contexts: similar vectors suggest *semantic relatedness* as a function of the distance between two words, capturing paradigmatic (e.g. synonymy) or syntagmatic relations (Pado, 2007). Vectors $\overrightarrow{h}$ are described by an adjacency matrix M, whose rows describe target words (h) and whose columns describe their corpus contexts. Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997), is then applied to M to acquire meaningful representations $\overrightarrow{h}$. LSA exploits the linear transformation called *Singular Value Decomposition* (SVD) and produces an approximation of the original matrix M, capturing (semantic) dependencies between context vectors. M is replaced by a lower dimensional matrix M_l, capturing the same statistical information in a new l-dimensional space, where each dimension is a linear combination of some of the original features (i.e. contexts). These derived features may be thought as artificial concepts, each one representing an emerging meaning component, as the linear combination of many different words.

In the argument classification task, the similarity between two argument heads h_1 and h_2 observed in FrameNet can be computed over $\overrightarrow{h_1}$ and $\overrightarrow{h_2}$. The model for a given frame element FE^k is built around the semantic heads h observed in the role FE^k: they form a set denoted by H^{FE^k}. These LSA vectors $\overrightarrow{h}$ express the individual annotated examples as they are immerse in the LSA space acquired from the unlabeled texts. Moreover, given FE^k, a model for each individual syntactic relation r (i.e. that links h labeled as FE^k to their corresponding predicates) is a partition of the set H^{FE^k} called $H_r^{FE^k}$, i.e. the subset of H^{FE^k} produced by examples of the relation r (e.g. Subj). Given the annotated sentence (2), we have that $report \in H^{\text{MEDIUM}}_{\text{SBJ}}$.

Role, FE^k	**Clusters of semantic heads**
MEDIUM	c_1: {*article, report, statement*} c_2: {*constitution, decree, rule*}
SPEAKER	c_3: {*brother, father, mother, sister* } c_4: {*biographer, philosopher, ...*} c_5: {*he, she, we, you*} c_6: {*friend*}
TOPIC	c_7: {*privilege, unresponsiveness*} c_8: {*pattern*}

Table 1: Clusters of semantic heads in the Subj position for the frame STATEMENT with σ = 0.5

As the LSA vectors $\overrightarrow{h}$ are available for the semantic heads h, a vector representation $\overrightarrow{FE^k}$ for the role FE^k can be obtained from the annotated data. However, one single vector is a too simplistic representation given the rich nature of semantic roles FE^k. In order to better represent FE^k, multiple regions in the semantic space are used. They are obtained by a clustering process applied to the set $H_r^{FE^k}$ according to the *Quality Threshold (QT)* algorithm (Heyer et al., 1999). QT is a generalization of k-mean where a variable number of clusters can be obtained. This number depends on the minimal value of intra-cluster similarity accepted by the algorithm and controlled by a parameter, σ: lower values of σ correspond to more heterogeneous (i.e. larger grain) clusters, while values close to 1 characterize stricter policies and more fine-grained results. Given a syntactic relation r, $C_r^{FE^k}$ denotes the clusters derived by QT clustering over $H_r^{FE^k}$. Each cluster $c \in C_r^{FE^k}$ is represented by a vector $\overrightarrow{c}$, computed as the geometric centroid of its semantic heads $h \in c$. For a frame F, clusters define a geometric model of every frame elements FE^k: it consists of centroids $\overrightarrow{c}$ with $c \subseteq H_r^{FE^k}$. Each c represents FE^k through a set of similar heads, as role fillers observed in FrameNet. Table 1 represents clusters for the heads $H^{FE^k}_{\text{Subj}}$ of the STATEMENT frame.

In argument classification we assume that the evoking predicate word for the frame F in an input sentence s is known. A sentence s can be seen as a sequence of role-relation pairs:

$s = \{(r_1, h_1), ..., (r_n, h_n)\}$ where the heads h_i are in the syntactic relation r_i with the underlying lexical unit of F.

For every head h in s, the vector $\overrightarrow{h}$ can be then used to estimate its similarity with the different candidate roles FE^k. Given the syntactic relation r, the clusters $c \in C_r^{FE^k}$ whose centroid vector $\vec{c}$ is closer to $\vec{h}$ are selected. $D_{r,h}$ is the set of the representations semantically related to h:

$$D_{r,h} = \bigcup_k \{c_{kj} \in C_r^{FE^k} | sim(h, c_{kj}) \geq \tau\} \quad (3)$$

where the similarity between the j-th cluster for the FE^k, i.e. $c_{kj} \in C_r^{FE^k}$, and h is the usual *cosine similarity*: $sim_{cos}(h, c_{kj}) = \frac{\overrightarrow{h} \cdot \overrightarrow{c}_{kj}}{\|\overrightarrow{h}\| \, \|\overrightarrow{c}_{kj}\|}$

Then, through a *k-nearest neighbours* (*k*-NN) strategy within $D_{r,h}$, the m clusters c_{kj} most similar to h are retained in the set $D_{r,h}^{(m)}$. A probabilistic preference for the role FE^k is estimated for h through a cluster-based voting scheme,

$$prob(FE^k | r, h) = \frac{|C_r^{FE^k} \cap D_{r,h}^{(m)}|}{|D_{r,h}^{(m)}|} \quad (4)$$

or, alternatively, an instance-based one over $D_{r,h}^{(m)}$:

$$prob(FE^k | r, h) = \frac{\sum_{c \in C_r^{FE^k} \cap D_{r,h}^{(m)}} |c|}{\sum_{c \in D_{r,h}^{(m)}} |c|} \quad (5)$$

In Fig. 1 the preference estimation for the incoming head $h = professor$ connected to a LU by the `Subj` relation is shown. Clusters for the heads in Table 1 are also reported. First, in the set of clusters whose similarity with $professor$ is higher than a threshold τ the $m = 5$ most similar clusters are selected. Accordingly, the preferences given by Eq. 4 are $prob(\text{SPEAKER}|\text{SBJ}, h) = 3/5$, $prob(\text{MEDIUM}|\text{SBJ}, h) = 2/5$ and $prob(\text{TOPIC}|\text{SBJ}, h) = 0$. The strategy modeled by Eq. 5 amplifies the role of larger clusters, e.g. $prob(\text{SPEAKER}|\text{SBJ}, h) = 9/14$ and $prob(\text{MEDIUM}|\text{SBJ}, h) = 5/14$. We call *Distributional*, the model that applies Eq. 5 to the source (r, h) arguments, by rejecting cases *only when* no information about the head h is available from the unlabeled corpus or no example of relation r for the role FE^k is available from the annotated corpus. Eq. 4 and 5 in fact do not cover all possible cases. Often the incoming head h or the relation r may be unavailable:

1. If the head h has never been met in the unlabeled corpus or the high grammatical ambiguity of the sentence does not allow to locate it reliably, Eq. 4 (or 5) should be backed off to a purely syntactic model, that is $prob(FE^k|r)$

2. If the relation r can not be properly located in s, h is also unknown: the prior probability of individual arguments, i.e. $prob(FE^k)$, is here employed.

Both $prob(FE^k|r)$ and $prob(FE^k)$ can be estimated from the training set and smoothing can be also applied[1]. A more robust argument preference function for all arguments $(r_i, h_i) \in s$ of the frame F is thus given by:

$$prob(FE^k | r_i, h_i) = \lambda_1 prob(FE^k | r_i, h_i) + \lambda_2 prob(FE^k | r_i) + \lambda_3 prob(FE^k) \quad (6)$$

where weights λ_1, λ_2, λ_3 can be heuristically assigned or estimated from the training set[2]. The resulting model is hereafter called *Backoff model*: although simply based on a single feature (i.e. the syntactic relation r), it accounts for information at different reliability degrees.

3.2 A Joint Model for Argument Classification

Eq. 6 defines roles preferences local to individual arguments (r_i, h_i). However, an argument frame is a joint structure, with strong dependencies between arguments. We thus propose to model the reranking phase (RR) as a HMM sequence labeling task. It defines a stochastic inference over multiple (locally justified) alternative sequences through a Hidden Markov Model (HMM). It infers the best sequence $FE^{(k_1,...,k_n)}$ over all the possible hidden state sequences (i.e. made by the target FE^{k_i}) given the observable emissions, i.e. the arguments (r_i, h_i). Viterbi inference is applied to build the best (role) interpretation for the input sentence.

Once Eq. 6 is available, the best frame element sequence $FE^{(\theta(1),...,\theta(n))}$ for the entire sentence s can be selected by defining the function $\theta(\cdot)$ that maps arguments $(r_i, h_i) \in s$ to frame elements FE^k:

$$\theta(i) = k \;\; s.t. \;\; FE^k \in F \quad (7)$$

[1] Lindstone smoothing was applied with $\delta = 1$.

[2] In each test discussed hereafter, λ_1, λ_2, λ_3 were assigned to .9,.09 and .01, in order to impose a strict priority to the model contributions.

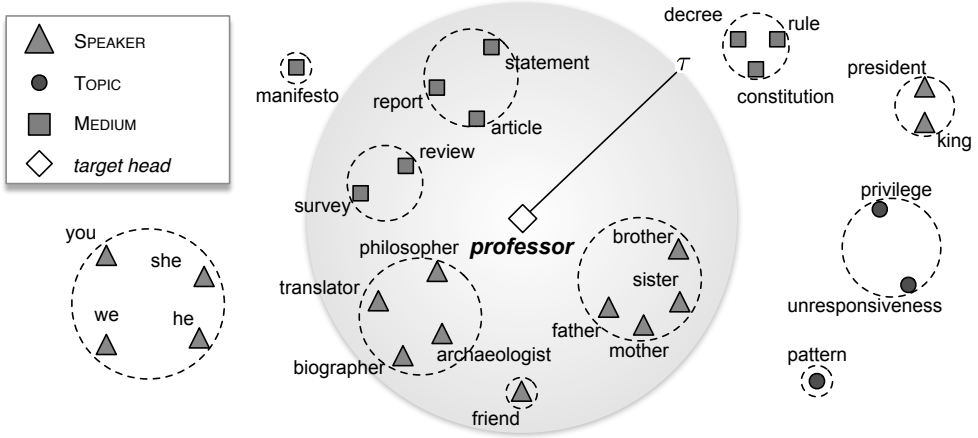


Figure 1: A k-NN approach to the role classification for $h_i = professor$

Notice that different transfer functions $\theta(\cdot)$ are usually possible. By computing their probability we can solve the SRL task by selecting the most likely interpretation, $\widehat{\theta}(\cdot)$, via $\text{argmax}_\theta\, P\big(\theta(\cdot) \mid s\big)$, as follows:

$$\widehat{\theta}(\cdot) = \underset{\theta}{\text{argmax}}\, P\big(s|\theta(\cdot)\big)P\big(\theta(\cdot)\big) \qquad (8)$$

In Eq. 8, the emission probability $P\big(s|\theta(\cdot)\big)$ and the transition probability $P\big(\theta(\cdot)\big)$ are explicit. Notice that the emission probability corresponds to an argument interpretation (e.g. Eq. 5) and it is assigned independently from the rest of the sentence. On the other hand, transition probabilities model role sequences and support the expectations about argument frames of a sentence.

The emission probability is approximated as:

$$P\big(s \mid \theta(1)\ldots\theta(n)\big) \approx \prod_{i=1}^{n} P(r_i, h_i \mid FE^{\theta(i)}) \qquad (9)$$

as it is made independent from previous states in a Viterbi path. Again the emission probability can be rewritten as:

$$P(r_i, h_i|FE^{\theta(i)}) = \frac{P(FE^{\theta(i)}|r_i, h_i)\; P(r_i, h_i)}{P(FE^{\theta(i)})} \qquad (10)$$

Since $P(r_i, h_i)$ does not depend on the role labeling, maximizing Eq. 10 corresponds to maximize:

$$\frac{P(FE^{\theta(i)}|r_i, h_i)}{P(FE^{\theta(i)})} \qquad (11)$$

whereas $P(FE^{\theta(i)}|r_i, h_i)$ is thus estimated through Eq. 6.

The transition probability, estimated through

$$P\big(\theta(1)\ldots\theta(n)\big) \approx \prod_{i=1}^{n} P\big(FE^{\theta(i)}|FE^{\theta(i-1)}, FE^{\theta(i-2)}\big) \qquad (12)$$

accounts FEs sequence via a *3*-gram model[3] .

4 Empirical Analysis

The aim of the evaluation is to measure the reachable accuracy of the simple model proposed and to compare its impact over in-domain and out-of-domain semantic role labeling tasks. In particular, we will evaluate the argument classification (AC) task in Section 4.2.

Experimental Set-Up. The in-domain test has been run over the FrameNet annotated corpus, derived from the British National Corpus (BNC). The splitting between train and test set is 90%-10% according to the same data set of (Johansson and Nugues, 2008b). In all experiments, the FrameNet 1.3 version and the dependency-based system using the LTH parser (Johansson and Nugues, 2008a) have been employed. Out-of-domain tests are run over the two training corpora as made available by the Semeval 2007 Task 19[4] (Baker et al., 2007): the Nuclear Threat Initiative (NTI) and the American National Corpus

[3]Two empty states are added at the beginning of any sequence. Moreover, Laplace smoothing was also applied to each estimator.

[4]The NTI and ANC annotated collections are downloadable at:
nlp.cs.swarthmore.edu/semeval/tasks/task19/data/train.tar.gz

	Corpus	Predicates	Arguments
training	FN-BNC	134,697	271,560
test			
in-domain	FN-BNC	14,952	30,173
out-of-domain	NTI	8,208	14,422
	ANC	760	1,389

Table 2: Training and Testing data sets

	Frames with a number of annotated examples					
Eq. - σ	>0	>100	>500	>1K	>3K	>5K
(5) - .85	**86.3**	86.5	87.2	**88.3**	85.9	82.0
(4) - .5	85.1	85.5	85.8	87.2	83.5	79.4
(4) - .1	84.5	84.8	85.1	86.5	83.0	78.7

Table 3: Accuracy on Arg classification tasks wrt different clustering policies

(ANC)[5]. Table 2 shows the predicates and arguments in each data set. All null-instantiated arguments were removed from the training and test sets.

Vectors $\vec{h}$ representing semantic heads have been computed according to the "dependency-based" vector space discussed in (Pado and Lapata, 2007). The entire BNC corpus has been parsed and the dependency graphs derived from individual sentences provided the basic observable contexts: every co-occurrence is thus syntactically justified by a dependency arc. The most frequent 30,000 basic features, i.e. (syntactic relation,lemma) pairs, have been used to build the matrix M, vector components corresponding to point-wise mutual information scores. Finally, the final space is obtained by applying the SVD reduction over M, with a dimensionality cut of $l = 250$.

In the evaluation of the AC task, accuracy is computed over the nodes of the dependency graph, in line with (Johansson and Nugues, 2008b) or (Coppola et al., 2009). Accordingly, also recall, precision and F-measure are reported on a *per node* basis, against the binary BD task or for the full $BD + AC$ chain.

4.1 The Role of Lexical Clustering

The first study aims at detecting the impact of different clustering policies on the resulting AC accuracy. Clustering, as discussed in Section 3.1, allows to generalize lexical information: similar heads within the latent semantic space are built from the annotated examples and they allow to predict the behavior of new unseen words as found in the test sentences. The system performances have been here measured under different clustering conditions, i.e. grains at which the clustering of annotated examples is applied. This grain is determined by the parameter σ of the applied Quality Threshold algorithm (Heyer et al., 1999). Notice that small values of σ imply large clusters, while if $\sigma \approx 1$ then many singleton clusters are promoted (i.e. one cluster for each example). By varying the threshold σ we thus account for prototype-based as well exemplar-based strategies, as discussed in (Erk, 2009).

We measured the performance on the argument classification tasks of different models obtained by combing different choices of σ with Eq. (4) or (5). Results are reported in Table 3. The leftmost column reports the different clustering settings, while in the remaining columns we see performances over test sentences related to different frames: we selected frames for which an increasing number of annotated examples are available: from all frames (for more than 0 examples) to the only frame (i.e. SELF_MOTION) that has more than 5,000 examples in our training data set.

The reported accuracies suggest that Eq. (5), promoting an example driven strategy, better captures the role preference, as it always outperforms alternative settings (i.e. more prototype oriented methods). It limits overgeneralization and promotes fine grained clusters. An interesting result is that a per-node accuracy of 86.3 (i.e. only 3 points under the state of-the art on the same data set, (Johansson and Nugues, 2008b)) is achieved. All the remaining tests have been run with the clustering configuration characterized by Eq. (5) and $\sigma = 0.85$.

4.2 Argument Classification Accuracy

In these experiments we evaluate the quality of the argument classification step against the lexical knowledge acquired from unlabeled texts and the reranking step. The accuracy reachable on the gold standard argument boundaries has been compared across several experimental settings. Two baseline systems have been obtained. The *Local Prior* model outputs the sequence that maximizes the prior probability locally to individual arguments. The *Global Prior* model is obtained by applying re-ranking (Section 3.2) to the best $n = 10$ candidates provided by the *Local Prior* model. Fi-

[5]Sentences whose arguments were not represented in the FrameNet training material were removed from all tests.

Model	FN-BNC	NTI	ANC
Local Prior	43.9	50.9	50.4
Global Prior	67.7 (+54.2%)	75.9 (+49.0%)	68.8 (+36.4%)
Distributional	81.1 (+19.8%)	82.3 (+8.4%)	69.7 (+1.3%)
Backoff	84.6 (+4.3%)	87.2 (+6.0%)	76.2 (+9.3%)
Backoff+HMMRR	86.3 (+2.0%)	90.5 (+3.8%)	79.9 (+5.0%)
(Johansson&Nugues, 2008)	89.9	71.1	-

Table 4: Accuracy of the Argument Classification task over the different corpora. In parenthesis the relative increment with respect to the immediately simpler model, previous row

nally, the application of the backoff strategies (as in Eq. 6) and the HMM-based reranking characterize the final two configurations. Table 4 reports the accuracy results obtained over the three corpora (defined as in Table 2): the accuracy scores are averaged over different values of m in Eq. 5, ranging from 3 to 30. In the in-domain scenario, i.e. the FN-BNC dataset reported in column 2, it is worth noticing that the proposed model, with backoff and global reranking, is quite effective with respect to the state-of-the-art.

Although results on the FN-BNC do not outperform the state-of-the-art for the FrameNet corpus, we still need to study the generalization capability of our SRL model in out-of-domain conditions. In a further experiment, we applied the same system, as trained over the FN-BNC data, to the other corpora, i.e. NTI and ANC, used entirely as test sets. Results, reported in column 3 and 4 of Table 4 and shown in Figure 2, confirm that no major drop in performance is observed. Notice how the positive impact of the backoff models and the HMM reranking policy is similarly reflected by all the collections. Moreover, the results on the NTI corpus are even better than those obtained on the BNC, with a resulting 90.5% accuracy on the AC task.

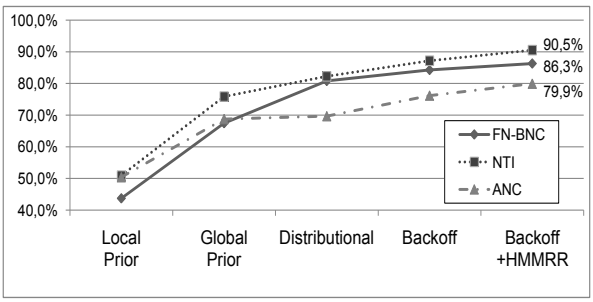


Figure 2: Accuracy of the AC task over different corpora

4.3 Discussion

The above empirical findings are relevant if compared with the outcome of a similar test on the NTI collection, discussed in (Johansson and Nugues, 2008b)[6]. There, under the same training conditions, a performance drop of about -19% is reported (from 89.9 to 71.1%) over gold standard argument boundaries. The model proposed in this paper exhibits no such drop in any collection (NTI and ANC). This seems to confirm the hypothesis that the model is able to properly generalize the required lexical information across different domains.

It is interesting to outline that the individual stages of the proposed model play different roles in the different domains, as Table 4 suggests. Although the positive contributions of the individual processing stages are uniformly confirmed, some differences can be outlined:

- The beneficial impact of the lexical information (i.e. the distributional model) applies differently across the different domains. The ANC domain seems not to significantly benefit when the distributional model (Eq. 5) is applied. Notice how Eq. 5 depends both from the evidence gathered in the corpus about lexical heads h as well as about the relation r. In ANC the percentage of times that the Eq. 5 is backed off against test instances (as h or r are not available from the training data) is twice as high as in the BNC-FN or in the NTI domain (i.e. 15.5 vs. 7.2 or 8.7, respectively). The different syntactic style of ANC seems thus the main responsible of the poor impact of distributional information, as it is often unapplicable to ANC test cases.

- The complexity of the three test sets is different, as the three plots show. The NTI col-

[6]Notice that in this paper only the training portion of the NTI data set is employed as reported in Table 2 and results are not directly comparable to (Johansson and Nugues, 2008b).

lections seems characterized by a lower level of complexity (see for example the accuracy of the *Local prior* model, that is about 51% as for the ANC). It then gets benefits from all the analysis stages, in particular the final HMM reranking. The BNC-FN test collection seems the most complex one, and the impact of the lexical information brought by the distributional model is here maximal. This is mainly due to the coherence between the distributions of lexical and grammatical phenomena in the test and training data.

- The role of HMM reranking is an effective way to compensate errors in the local argument classifications for all the three domains. However, it is particularly effective for the outside domain cases, while, in the BNC corpus, it produces just a small improvement instead (i.e. +2%, as shown in Table 4). It is worth noticing that the average length of the sentences in the BNC test collection is about 23 words per sentence, while it is higher for the NTI and ANC data sets (i.e. 34 and 31, respectively). It seems that the HMM model well captures some information on the global semantic structure of a sentence: this is helpful in cases where errors in the grammatical recognition (of individual arguments or at sentence level) are more frequent and afflict the local distributional model. The more complex is the syntax of a corpus (e.g. in the NTI and ANC data sets), the higher seems the impact of the reranking phase.

The significant performance of the AC model here presented suggest to test it when integrated within a full SRL architecture. Table 5 reports the results of the processing cascade over three collections. Results on the Boundary Detection BD task are obtained by training an SVM model on the same feature set presented in (Johansson and Nugues, 2008b) and are slightly below the state-of-the art BD accuracy reported in (Coppola et al., 2009). However, the accuracy of the complete $BD + AC + RR$ chain (i.e. 68%) improves the corresponding results of (Coppola et al., 2009). Given the relatively simple feature set adopted here, this result is very significant as for its resulting efficiency. The overall BD recognition process is, on a standard architecture, performed at about 6.74 sentences per second, that is basically the same as the time needed for applying the entire $BD + AC + RR$ chain, i.e. 6.21 sentence per second.

Corpus	Eval. Setting	Recall	Precision	F1
BNC	BD	72.6	85.1	78.4
	BD+AC+RR	62.6	74.5	**68.0**
NTI	BD	63.9	80.0	71.0
	BD+AC+RR	56.7	72.1	63.5
ANC	BD	64.0	81.5	71.7
	BD+AC+RR	47.4	62.5	53.9

Table 5: Accuracy of the full cascade of the SRL system over three domain

5 Conclusions

In this paper, a distributional approach for acquiring a semi-supervised model of argument classification (AC) preferences has been proposed. It aims at improving the generalization capability of the inductive SRL approach by reducing the complexity of the employed grammatical features and through a distributional representation of lexical features. The obtained results are close to the state-of-art in FrameNet semantic parsing. State of the art accuracy is obtained instead in out-of-domain experiments. The model seems to capitalize from simple methods of lexical modeling (i.e. the estimation of lexico-grammatical preferences through distributional analysis over unlabeled data), estimation (through syntactic or lexical back-off where necessary) and reranking. The result is an accurate and highly portable SRL cascade. Experiments on the integrated SRL architecture (i.e. $BD + AC + RR$ chain) show that state-of-art accuracy (i.e. 68%) can be obtained on raw texts. This result is also very significant as for the achieved efficiency. The system is able to apply the entire $BD + AC + RR$ chain at a speed of 6.21 sentences per second. This significant efficiency confirms the applicability of the SRL approach proposed here in large scale NLP applications. Future work will study the application of the flexible SRL method proposed to other languages, for which less resources are available and worst training conditions are the norm. Moreover, dimensionality reduction methods alternative to LSA, as currently studied on semi-supervised spectral learning (Johnson and Zhang, 2008), will be experimented.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL*, Montreal, Canada.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. Semeval-2007 task 19: Frame semantic structure extraction. In *Proceedings of SemEval-2007*, pages 99–104, Prague, Czech Republic, June. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proc. of CoNLL-2005*, pages 152–164, Ann Arbor, Michigan, June.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *In Proceedings of ICML '08*, pages 160–167, New York, NY, USA. ACM.

Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Shallow semantic parsing for spoken language understanding. In *Proceedings of NAACL '09*, pages 85–88, Morristown, NJ, USA.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Morristown, NJ, USA. Association for Computational Linguistics.

Katrin Erk and Sebastian Pado. 2006. Shalmaneser - a flexible toolbox for semantic role assignment. In *Proceedings of LREC 2006*, Genoa, Italy.

Katrin Erk. 2009. Representing words as regions in vector space. In *In Proceedings of CoNLL '09*, pages 57–65, Morristown, NJ, USA. Association for Computational Linguistics.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica*, 4(2):222–254.

Hagen Fürstenau and Mirella Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *In Proceedings of EMNLP '09*, pages 11–20, Morristown, NJ, USA.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Yoav Goldberg and Michael Elhadad. 2009. On the role of lexical features in sequence labeling. In *In Proceedings of EMNLP '09*, pages 1142–1151, Singapore, August. Association for Computational Linguistics.

L.J. Heyer, S. Kruglyak, and S. Yooseph. 1999. Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research*, (9):1106–1115.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of CoNLL-2008*, Manchester, UK, August 16-17.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*, Manchester, UK, August 18-22.

Rie Johnson and Tong Zhang. 2008. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288.

Tom Landauer and Sue Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104.

A. Moschitti, D. Pighin, and R. Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34.

Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Sebastian Pado. 2007. *Cross-Lingual Annotation Projection Models for Role-Semantic Information*. Ph.D. thesis, Saarland University.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1), March.

Sameer S. Pradhan, Wayne Ward, and James H. Martin. 2008. Towards robust semantic role labeling. *Comput. Linguist.*, 34(2):289–310.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 127–132, Morristown, NJ, USA. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Comput. Linguist.*, 34(2):161–191.

A Bayesian Method for Robust Estimation of Distributional Similarities

Jun'ichi Kazama Stijn De Saeger Kow Kuroda
Masaki Murata† Kentaro Torisawa
Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289 Japan
{kazama, stijn, kuroda, torisawa}@nict.go.jp
†Department of Information and Knowledge Engineering
Faculty/Graduate School of Engineering, Tottori University
4-101 Koyama-Minami, Tottori, 680-8550 Japan*
murata@ike.tottori-u.ac.jp

Abstract

Existing word similarity measures are not robust to data sparseness since they rely only on the point estimation of words' context profiles obtained from a limited amount of data. This paper proposes a Bayesian method for robust distributional word similarities. The method uses a distribution of context profiles obtained by Bayesian estimation and takes the expectation of a base similarity measure under that distribution. When the context profiles are multinomial distributions, the priors are Dirichlet, and the base measure is the Bhattacharyya coefficient, we can derive an analytical form that allows efficient calculation. For the task of word similarity estimation using a large amount of Web data in Japanese, we show that the proposed measure gives better accuracies than other well-known similarity measures.

1 Introduction

The semantic similarity of words is a long-standing topic in computational linguistics because it is theoretically intriguing and has many applications in the field. Many researchers have conducted studies based on the distributional hypothesis (Harris, 1954), which states that words that occur in the same contexts tend to have similar meanings. A number of semantic similarity measures have been proposed based on this hypothesis (Hindle, 1990; Grefenstette, 1994; Dagan et al., 1994; Dagan et al., 1995; Lin, 1998; Dagan et al., 1999).

In general, most semantic similarity measures have the following form:

$$sim(w_1, w_2) = g(v(w_1), v(w_2)), \qquad (1)$$

where $v(w_i)$ is a vector that represents the contexts in which w_i appears, which we call a *context profile* of w_i. The function g is a function on these context profiles that is expected to produce good similarities. Each dimension of the vector corresponds to a context, f_k, which is typically a neighboring word or a word having dependency relations with w_i in a corpus. Its value, $v_k(w_i)$, is typically a co-occurrence frequency $c(w_i, f_k)$, a conditional probability $p(f_k|w_i)$, or point-wise mutual information (PMI) between w_i and f_k, which are all calculated from a corpus. For g, various works have used the cosine, the Jaccard coefficient, or the Jensen-Shannon divergence is utilized, to name only a few measures.

Previous studies have focused on how to devise good contexts and a good function g for semantic similarities. On the other hand, our approach in this paper is to estimate context profiles ($v(w_i)$) robustly and thus to estimate the similarity robustly. The problem here is that $v(w_i)$ is computed from a corpus of limited size, and thus inevitably contains uncertainty and sparseness. The guiding intuition behind our method is as follows. *All other things being equal, the similarity with a more frequent word should be larger, since it would be more reliable.* For example, if $p(f_k|w_1)$ and $p(f_k|w_2)$ for two given words w_1 and w_2 are equal, but w_1 is more frequent, we would expect that $sim(w_0, w_1) > sim(w_0, w_2)$.

In the NLP field, data sparseness has been recognized as a serious problem and tackled in the context of language modeling and supervised machine learning. However, to our knowledge, there

*The work was done while the author was at NICT.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 247–256,
Uppsala, Sweden, 11-16 July 2010.

has been no study that seriously dealt with data sparseness in the context of semantic similarity calculation. The data sparseness problem is usually solved by smoothing, regularization, margin maximization and so on (Chen and Goodman, 1998; Chen and Rosenfeld, 2000; Cortes and Vapnik, 1995). Recently, the Bayesian approach has emerged and achieved promising results with a clearer formulation (Teh, 2006; Mochihashi et al., 2009).

In this paper, we apply the Bayesian framework to the calculation of distributional similarity. The method is straightforward: Instead of using the point estimation of $v(w_i)$, we first estimate the distribution of the context profile, $p(v(w_i))$, by Bayesian estimation and then take the expectation of the original similarity under this distribution as follows:

$$\begin{aligned} & sim_b(w_1, w_2) \\ & = E[sim(w_1, w_2)]_{\{p(v(w_1)),p(v(w_2))\}} \\ & = E[g(v(w_1), v(w_2))]_{\{p(v(w_1)),p(v(w_2))\}}. \end{aligned} \quad (2)$$

The uncertainty due to data sparseness is represented by $p(v(w_i))$, and taking the expectation enables us to take this into account. The Bayesian estimation usually gives diverging distributions for infrequent observations and thus decreases the expectation value as expected.

The Bayesian estimation and the expectation calculation in Eq. 2 are generally difficult and usually require computationally expensive procedures. Since our motivation for this research is to calculate good semantic similarities for a large set of words (e.g., one million nouns) and apply them to a wide range of NLP tasks, such costs must be minimized.

Our technical contribution in this paper is to show that in the case where the context profiles are multinomial distributions, the priors are Dirichlet, and the base similarity measure is the Bhattacharyya coefficient (Bhattacharyya, 1943), we can derive an analytical form for Eq. 2, that enables efficient calculation (with some implementation tricks).

In experiments, we estimate semantic similarities using a large amount of Web data in Japanese and show that the proposed measure gives better word similarities than a non-Bayesian Bhattacharyya coefficient or other well-known similarity measures such as Jensen-Shannon divergence and the cosine with PMI weights.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the Bayesian estimation and the Bhattacharyya coefficient. Section 3 proposes our new Bayesian Bhattacharyya coefficient for robust similarity calculation. Section 4 mentions some implementation issues and the solutions. Then, Section 5 reports the experimental results.

2 Background

2.1 Bayesian estimation with Dirichlet prior

Assume that we estimate a probabilistic model for the observed data D, $p(D|\phi)$, which is parameterized with parameters ϕ. In the maximum likelihood estimation (MLE), we find the point estimation $\phi^* = \mathrm{argmax}_\phi p(D|\phi)$. For example, we estimate $p(f_k|w_i)$ as follows with MLE:

$$p(f_k|w_i) = c(w_i, f_k) / \sum_k c(w_i, f_k). \quad (3)$$

On the other hand, the objective of the Bayesian estimation is to find the distribution of ϕ given the observed data D, i.e., $p(\phi|D)$, and use it in later processes. Using Bayes' rule, this can also be viewed as:

$$p(\phi|D) = \frac{p(D|\phi)p_{prior}(\phi)}{p(D)}. \quad (4)$$

$p_{prior}(\phi)$ is a prior distribution that represents the plausibility of each ϕ based on the prior knowledge. In this paper, we consider the case where ϕ is a multinomial distribution, i.e., $\sum_k \phi_k = 1$, that models the process of choosing one out of K choices. Estimating a conditional probability distribution $\phi_k = p(f_k|w_i)$ as a context profile for each w_i falls into this case. In this paper, we also assume that the prior is the Dirichlet distribution, $Dir(\alpha)$. The Dirichlet distribution is defined as follows.

$$Dir(\phi|\alpha) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \phi_k^{\alpha_k - 1}. \quad (5)$$

$\Gamma(.)$ is the Gamma function. The Dirichlet distribution is parametrized by hyperparameters $\alpha_k (> 0)$.

It is known that $p(\phi|D)$ is also a Dirichlet distribution for this simplest case, and it can be analytically calculated as follows.

$$p(\phi|D) = Dir(\phi|\{\alpha_k + c(k)\}), \quad (6)$$

where $c(k)$ is the frequency of choice k in data D. For example, $c(k) = c(w_i, f_k)$ in the estimation of $p(f_k|w_i)$. This is very simple: we just need to add the observed counts to the hyperparameters.

2.2 Bhattacharyya coefficient

When the context profiles are probability distributions, we usually utilize the measures on probability distributions such as the Jensen-Shannon (JS) divergence to calculate similarities (Dagan et al., 1994; Dagan et al., 1997). The JS divergence is defined as follows.

$$JS(p_1||p_2) = \frac{1}{2}(KL(p_1||p_{avg}) + KL(p_2||p_{avg})),$$

where $p_{avg} = \frac{p_1+p_2}{2}$ is a point-wise average of p_1 and p_2 and $KL(.)$ is the Kullback-Leibler divergence. Although we found that the JS divergence is a good measure, it is difficult to derive an efficient calculation of Eq. 2, even in the Dirichlet prior case.[1]

In this study, we employ the Bhattacharyya coefficient (Bhattacharyya, 1943) (BC for short), which is defined as follows:

$$BC(p_1, p_2) = \sum_{k=1}^{K} \sqrt{p_{1k} \times p_{2k}}.$$

The BC is also a similarity measure on probability distributions and is suitable for our purposes as we describe in the next section. Although BC has not been explored well in the literature on distributional word similarities, it is also a good similarity measure as the experiments show.

3 Method

In this section, we show that if our base similarity measure is BC and the distributions under which we take the expectation are Dirichlet distributions, then Eq. 2 also has an analytical form, allowing efficient calculation.

Here, we calculate the following value given two Dirichlet distributions:

$$BC_b(p_1, p_2) = E[BC(p_1, p_2)]_{\{Dir(p_1|\alpha'), Dir(p_2|\beta')\}}$$
$$= \iint_{\triangle\times\triangle} Dir(p_1|\alpha')Dir(p_2|\beta')BC(p_1, p_2)dp_1dp_2.$$

After several derivation steps (see Appendix A), we obtain the following analytical solution for the above:

$$= \frac{\Gamma(\alpha'_0)\Gamma(\beta'_0)}{\Gamma(\alpha'_0 + \frac{1}{2})\Gamma(\beta'_0 + \frac{1}{2})} \sum_{k=1}^{K} \frac{\Gamma(\alpha'_k + \frac{1}{2})\Gamma(\beta'_k + \frac{1}{2})}{\Gamma(\alpha'_k)\Gamma(\beta'_k)}, \quad (7)$$

where $\alpha'_0 = \sum_k \alpha'_k$ and $\beta'_0 = \sum_k \beta'_k$. Note that with the Dirichlet prior, $\alpha'_k = \alpha_k + c(w_1, f_k)$ and $\beta'_k = \beta_k + c(w_2, f_k)$, where α_k and β_k are the hyperparameters of the priors of w_1 and w_2, respectively.

To put it all together, we can obtain a new Bayesian similarity measure on words, which can be calculated only from the hyperparameters for the Dirichlet prior, α and β, and the observed counts $c(w_i, f_k)$. It is written as follows.

$$BC_b(w_1, w_2) = \quad (8)$$
$$\frac{\Gamma(\alpha_0 + a_0)\Gamma(\beta_0 + b_0)}{\Gamma(\alpha_0 + a_0 + \frac{1}{2})\Gamma(\beta_0 + b_0 + \frac{1}{2})} \times$$
$$\sum_{k=1}^{K} \frac{\Gamma(\alpha_k + c(w_1, f_k) + \frac{1}{2})\Gamma(\beta_k + c(w_2, f_k) + \frac{1}{2})}{\Gamma(\alpha_k + c(w_1, f_k))\Gamma(\beta_k + c(w_2, f_k))},$$

where $a_0 = \sum_k c(w_1, f_k)$ and $b_0 = \sum_k c(w_2, f_k)$. We call this new measure *the Bayesian Bhattacharyya coefficient* (BC_b for short). For simplicity, we assume $\alpha_k = \beta_k = \alpha$ in this paper.

We can see that BC_b actually encodes our guiding intuition. Consider four words, w_0, w_1, w_2, and w_4, for which we have $c(w_0, f_1) = 10$, $c(w_1, f_1) = 2$, $c(w_2, f_1) = 10$, and $c(w_3, f_1) = 20$. They have counts only for the first dimension, i.e., they have the same context profile: $p(f_1|w_i) = 1.0$, when we employ MLE. When $K = 10,000$ and $\alpha_k = 1.0$, the Bayesian similarity between these words is calculated as

$$BC_b(w_0, w_1) = 0.785368$$
$$BC_b(w_0, w_2) = 0.785421$$
$$BC_b(w_0, w_3) = 0.785463$$

We can see that similarities are different according to the number of observations, as expected. Note that the non-Bayesian BC will return the same value, 1.0, for all cases. Note also that $BC_b(w_0, w_0) = 0.78542$ if we use Eq. 8, meaning that the self-similarity might not be the maximum. This is conceptually strange, although not a serious problem since we hardly use $sim(w_i, w_i)$ in practice. If we want to fix this, we can use the special definition: $BC_b(w_i, w_i) \equiv 1$. This is equivalent to using $sim_b(w_i, w_i) = E[sim(w_i, w_i)]_{\{p(v(w_i))\}} = 1$ only for this case.

[1] A naive but general way might be to draw samples of $v(w_i)$ from $p(v(w_i))$ and approximate the expectation using these samples. However, such a method will be slow.

4 Implementation Issues

Although we have derived the analytical form (Eq. 8), there are several problems in implementing robust and efficient calculations.

First, the Gamma function in Eq. 8 overflows when the argument is larger than 170. In such cases, a commonly used way is to work in the logarithmic space. In this study, we utilize the "log Gamma" function: $ln\Gamma(x)$, which returns the logarithm of the Gamma function directly without the overflow problem.[2]

Second, the calculation of the log Gamma function is heavier than operations such as simple multiplication, which is used in existing measures. In fact, the log Gamma function is implemented using an iterative algorithm such as the Lanczos method. In addition, according to Eq. 8, it seems that we have to sum up the values for all k, because even if $c(w_i, f_k)$ is zero the value inside the summation will not be zero. In the existing measures, it is often the case that we only need to sum up for k where $c(w_i, f_k) > 0$. Because $c(w_i, f_k)$ is usually sparse, that technique speeds up the calculation of the existing measures drastically and makes it practical.

In this study, the above problem is solved by pre-computing the required log Gamma values, assuming that we calculate similarities for a large set of words, and pre-computing default values for cases where $c(w_i, f_k) = 0$. The following values are pre-computed once at the start-up time.

For each word:

(A) $ln\Gamma(\alpha_0 + a_0) - ln\Gamma(\alpha_0 + a_0 + \frac{1}{2})$

(B) $ln\Gamma(\alpha_k + c(w_i, f_k)) - ln\Gamma(\alpha_k + c(w_i, f_k) + \frac{1}{2})$ for all k where $c(w_i, f_k) > 0$

(C) $-\exp(2(ln\Gamma(\alpha_k + \frac{1}{2}) - ln\Gamma(\alpha_k)))) + \exp(ln\Gamma(\alpha_k + c(w_i, f_k)) - ln\Gamma(\alpha_k + c(w_i, f_k) + \frac{1}{2}) + ln\Gamma(\alpha_k + \frac{1}{2}) - ln\Gamma(\alpha_k))$ for all k where $c(w_i, f_k) > 0$;

For each k:
(D): $\exp(2(ln\Gamma(\alpha_k + \frac{1}{2}))$.

In the calculation of $BC_b(w_1, w_2)$, we first assume that all $c(w_i, f_k) = 0$ and set the output variable to the default value. Then, we iterate over the sparse vectors $c(w_1, f_k)$ and $c(w_2, f_k)$. If $c(w_1, f_k) > 0$ and $c(w_2, f_k) = 0$ (and vice versa), we update the output variable just by adding (C). If $c(w_1, f_k) > 0$ and $c(w_2, f_k) > 0$, we update the output value using (B), (D) and one additional $\exp(.)$ operation. With this implementation, we can make the computation of BC_b practically as fast as using other measures.

[2] We used the GNU Scientific Library (GSL) (www.gnu.org/software/gsl/), which implements this function.

5 Experiments

5.1 Evaluation setting

We evaluated our method in the calculation of similarities between nouns in Japanese.

Because human evaluation of word similarities is very difficult and costly, we conducted automatic evaluation in the set expansion setting, following previous studies such as Pantel et al. (2009).

Given a word set, which is expected to contain similar words, we assume that a good similarity measure should output, for each word in the set, the other words in the set as similar words. For given word sets, we can construct input-and-answers pairs, where the answers for each word are the other words in the set the word appears in.

We output a ranked list of 500 similar words for each word using a given similarity measure and checked whether they are included in the answers. This setting could be seen as document retrieval, and we can use an evaluation measure such as the mean of the precision at top T (MP@T) or the mean average precision (MAP). For each input word, P@T (precision at top T) and AP (average precision) are defined as follows.

$$\text{P@}T = \frac{1}{T}\sum_{i=1}^{T}\delta(w_i \in ans),$$

$$\text{AP} = \frac{1}{R}\sum_{i=1}^{N}\delta(w_i \in ans)\text{P@}i.$$

$\delta(w_i \in ans)$ returns 1 if the output word w_i is in the answers, and 0 otherwise. N is the number of outputs and R is the number of the answers. MP@T and MAP are the averages of these values over all input words.

5.2 Collecting context profiles

Dependency relations are used as context profiles as in Kazama and Torisawa (2008) and Kazama et al. (2009). From a large corpus of Japanese Web documents (Shinzato et al., 2008) (100 million

documents), where each sentence has a dependency parse, we extracted noun-verb and noun-noun dependencies with relation types and then calculated their frequencies in the corpus. If a noun, n, depends on a word, w, with a relation, r, we collect a dependency pair, $(n, \langle w, r \rangle)$. That is, a context f_k, is $\langle w, r \rangle$ here.

For noun-verb dependencies, postpositions in Japanese represent relation types. For example, we extract a dependency relation (ワイン, ⟨ 買う, を ⟩) from the sentence below, where a postposition "を (wo)" is used to mark the verb object.

ワイン (wine) を (wo) 買う (buy) (≈ buy a wine)

Note that we leave various auxiliary verb suffixes, such as "れる (reru)," which is for passivization, as a part of w, since these greatly change the type of n in the dependent position.

As for noun-noun dependencies, we considered expressions of type "n_1 の n_2" (≈ "n_2 of n_1") as dependencies $(n_1, \langle n_2, \text{の} \rangle)$.

We extracted about 470 million unique dependencies from the corpus, containing 31 million unique nouns (including compound nouns as determined by our filters) and 22 million unique contexts, f_k. We sorted the nouns according to the number of unique co-occurring contexts and the contexts according to the number of unique co-occurring nouns, and then we selected the top one million nouns and 100,000 contexts. We used only 260 million dependency pairs that contained both the selected nouns and the selected contexts.

5.3 Test sets

We prepared three test sets as follows.

Set "A" and "B": Thesaurus siblings We considered that words having a common hypernym (i.e., siblings) in a manually constructed thesaurus could constitute a similar word set. We extracted such sets from a Japanese dictionary, EDR (V3.0) (CRL, 2002), which contains concept hierarchies and the mapping between words and concepts. The dictionary contains 304,884 nouns. In all, 6,703 noun sibling sets were extracted with the average set size of 45.96. We randomly chose 200 sets each for sets "A" and "B." Set "A" is a development set to tune the value of the hyperparameters and "B" is for the validation of the parameter tuning.

Set "C": Closed sets Murata et al. (2004) constructed a dataset that contains several closed word sets such as the names of countries, rivers, sumo wrestlers, etc. We used all of the 45 sets that are marked as "complete" in the data, containing 12,827 unique words in total.

Note that we do not deal with ambiguities in the construction of these sets as well as in the calculation of similarities. That is, a word can be contained in several sets, and the answers for such a word is the union of the words in the sets it belongs to (excluding the word itself).

In addition, note that the words in these test sets are different from those of our one-million-word vocabulary. We filtered out the words that are not included in our vocabulary and removed the sets with size less than 2 after the filtering.

Set "A" contained 3,740 words that are actually evaluated, with about 115 answers on average, and "B" contained 3,657 words with about 65 answers on average. Set "C" contained 8,853 words with about 1,700 answers on average.

5.4 Compared similarity measures

We compared our Bayesian Bhattacharyya similarity measure, BC_b, with the following similarity measures.

JS Jensen-Shannon divergence between $p(f_k|w_1)$ and $p(f_k|w_2)$ (Dagan et al., 1994; Dagan et al., 1999).

PMI-cos The cosine of the context profile vectors, where the k-th dimension is the pointwise mutual information (PMI) between w_i and f_k defined as: $PMI(w_i, f_k) = \log \frac{p(w_i, f_k)}{p(w_i)p(f_k)}$ (Pantel and Lin, 2002; Pantel et al., 2009).[3]

Cls-JS Kazama et al. (2009) proposed using the Jensen-Shannon divergence between hidden class distributions, $p(c|w_1)$ and $p(c|w_2)$, which are obtained by using an EM-based clustering of dependency relations with a model $p(w_i, f_k) = \sum_c p(w_i|c)p(f_k|c)p(c)$ (Kazama and Torisawa, 2008). In order to

[3]We did not use the discounting of the PMI values described in Pantel and Lin (2002).

alleviate the effect of local minima of the EM clustering, they proposed averaging the similarities by several different clustering results, which can be obtained by using different initial parameters. In this study, we combined two clustering results (denoted as "s1+s2" in the results), each of which ("s1" and "s2") has 2,000 hidden classes.[4] We included this method since clustering can be regarded as another way of treating data sparseness.

BC The Bhattacharyya coefficient (Bhattacharyya, 1943) between $p(f_k|w_1)$ and $p(f_k|w_2)$. This is the baseline for BC$_b$.

BC$_a$ The Bhattacharyya coefficient with absolute discounting. In calculating $p(f_k|w_i)$, we subtract the discounting value, α, from $c(w_i, f_k)$ and equally distribute the residual probability mass to the contexts whose frequency is zero. This is included as an example of naive smoothing methods.

Since it is very costly to calculate the similarities with all of the other words (one million in our case), we used the following approximation method that exploits the sparseness of $c(w_i, f_k)$. Similar methods were used in Pantel and Lin (2002), Kazama et al. (2009), and Pantel et al. (2009) as well. For a given word, w_i, we sort the contexts in descending order according to $c(w_i, f_k)$ and retrieve the top-L contexts.[5] For each selected context, we sort the words in descending order according to $c(w_i, f_k)$ and retrieve the top-M words ($L = M = 1600$).[6] We merge all of the words above as candidate words and calculate the similarity only for the candidate words. Finally, the top 500 similar words are output.

Note also that we used modified counts, $log(c(w_i, f_k)) + 1$, instead of raw counts, $c(w_i, f_k)$, with the intention of alleviating the effect of strangely frequent dependencies, which can be found in the Web data. In preliminary experiments, we observed that this modification improves the quality of the top 500 similar words as reported in Terada et al. (2004) and Kazama et al. (2009).

[4] In the case of EM clustering, the number of unique contexts, f_k, was also set to one million instead of 100,000, following Kazama et al. (2009).

[5] It is possible that the number of contexts with non-zero counts is less than L. In that case, all of the contexts with non-zero counts are used.

[6] Sorting is performed only once in the initialization step.

Table 1: Performance on siblings (Set A).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.0299	0.197	0.122	0.0990	0.0792
PMI-cos	0.0332	0.195	0.124	0.0993	0.0798
Cls-JS (s1)	0.0319	0.195	0.122	0.0988	0.0796
Cls-JS (s2)	0.0295	0.198	0.122	0.0981	0.0786
Cls-JS (s1+s2)	0.0333	0.206	0.129	0.103	0.0841
BC	0.0334	0.211	0.131	0.106	0.0854
BC$_b$ (0.0002)	0.0345	0.223	0.138	0.109	0.0873
BC$_b$ (0.0016)	**0.0356**	**0.242**	**0.148**	**0.119**	**0.0955**
BC$_b$ (0.0032)	0.0325	0.223	0.137	0.111	0.0895
BC$_a$ (0.0016)	0.0337	0.212	0.133	0.107	0.0863
BC$_a$ (0.0362)	0.0345	0.221	0.136	0.110	0.0890
BC$_a$ (0.1)	0.0324	0.214	0.128	0.101	0.0825
without $log(c(w_i, f_k)) + 1$ modification					
JS	0.0294	0.197	0.116	0.0912	0.0712
PMI-cos	0.0342	0.197	0.125	0.0987	0.0793
BC	0.0296	0.201	0.118	0.0915	0.0721

As for BC$_b$, we assumed that all of the hyperparameters had the same value, i.e., $\alpha_k = \alpha$. It is apparent that an excessively large α is not appropriate because it means ignoring observations. Therefore, α must be tuned. The discounting value of BC$_a$ is also tuned.

5.5 Results

Table 1 shows the results for Set A. The MAP and the MPs at the top 1, 5, 10, and 20 are shown for each similarity measure. As for BC$_b$ and BC$_a$, the results for the tuned and several other values for α are shown. Figure 1 shows the parameter tuning for BC$_b$ with MAP as the y-axis (results for BC$_a$ are shown as well). Figure 2 shows the same results with MPs as the y-axis. The MAP and MPs showed a correlation here. From these results, we can see that BC$_b$ surely improves upon BC, with 6.6% improvement in MAP and 14.7% improvement in MP@1 when $\alpha = 0.0016$. BC$_b$ achieved the best performance among the compared measures with this setting. The absolute discounting, BC$_a$, improved upon BC as well, but the improvement was smaller than with BC$_b$. Table 1 also shows the results for the case where we did not use the log-modified counts. We can see that this modification gives improvements (though slight or unclear for PMI-cos).

Because tuning hyperparameters involves the possibility of overfitting, its robustness should be assessed. We checked whether the tuned α with Set A works well for Set B. The results are shown in Table 2. We can see that the best α ($= 0.0016$) found for Set A works well for Set B as well. That is, the tuning of α as above is not unrealistic in

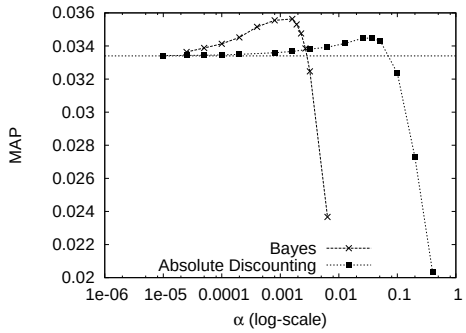


Figure 1: Tuning of α (MAP). The dashed horizontal line indicates the score of BC.

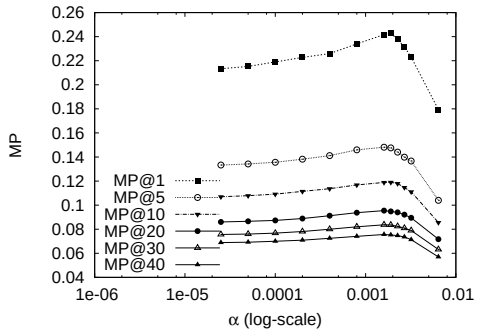


Figure 2: Tuning of α (MP).

practice because it seems that we can tune it robustly using a small subset of the vocabulary as shown by this experiment.

Next, we evaluated the measures on Set C, i.e., the closed set data. The results are shown in Table 3. For this set, we observed a tendency that is different from Sets A and B. Cls-JS showed a particularly good performance. BC_b surely improves upon BC. For example, the improvement was 7.5% for MP@1. However, the improvement in MAP was slight, and MAP did not correlate well with MPs, unlike in the case of Sets A and B.

We thought one possible reason is that the number of outputs, 500, for each word was not large enough to assess MAP values correctly because the average number of answers is 1,700 for this dataset. In fact, we could output more than 500 words if we ignored the cost of storage. Therefore, we also included the results for the case where $L = M = 3600$ and $N = 2,000$. Even with this setting, however, MAP did not correlate well with MPs.

Although Cls-JS showed very good performance for Set C, note that the EM clustering is very time-consuming (Kazama and Torisawa, 2008), and it took about one week with 24 CPU cores to get one clustering result in our computing environment. On the other hand, the preparation for our method requires just an hour with a single core.

Table 2: Performance on siblings (Set B).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.0265	0.208	0.116	0.0855	0.0627
PMI-cos	0.0283	0.203	0.116	0.0871	0.0660
Cls-JS (s1+s2)	0.0274	0.194	0.115	0.0859	0.0643
BC	0.0295	0.223	0.124	0.0922	0.0693
BC_b (0.0002)	0.0301	0.225	0.128	0.0958	0.0718
BC_b (0.0016)	**0.0313**	**0.246**	**0.135**	**0.103**	**0.0758**
BC_b (0.0032)	0.0279	0.228	0.127	0.0938	0.0698
BC_a (0.0016)	0.0297	0.223	0.125	0.0934	0.0700
BC_a (0.0362)	0.0298	0.223	0.125	0.0934	0.0705
BC_a (0.01)	0.0300	0.224	0.126	0.0949	0.0710

Table 3: Performance on closed-sets (Set C).

Measure	MAP	MP			
		@1	@5	@10	@20
JS	0.127	0.607	0.582	0.566	0.544
PMI-cos	0.124	0.531	0.519	0.508	0.493
Cls-JS (s1)	0.125	0.589	0.566	0.548	0.525
Cls-JS (s2)	0.137	0.608	0.592	0.576	0.554
Cls-JS (s1+s2)	**0.152**	0.638	**0.617**	**0.603**	**0.583**
BC	0.131	0.602	0.579	0.565	0.545
BC_b (0.0004)	0.133	0.636	0.605	0.587	0.563
BC_b (0.0008)	0.131	**0.647**	0.615	0.594	0.568
BC_b (0.0016)	0.126	0.644	0.615	0.593	0.564
BC_b (0.0032)	0.107	0.573	0.556	0.529	0.496
$L = M = 3200$ and $N = 2000$					
JS	0.165	0.605	0.580	0.564	0.543
PMI-cos	0.165	0.530	0.517	0.507	0.492
Cls-JS (s1+s2)	0.209	0.639	0.618	0.603	0.584
BC	0.168	0.600	0.577	0.562	0.542
BC_b (0.0004)	0.170	0.635	0.604	0.586	0.562
BC_b (0.0008)	0.168	0.647	0.615	0.594	0.568
BC_b (0.0016)	0.161	0.644	0.615	0.593	0.564
BC_b (0.0032)	0.140	0.573	0.556	0.529	0.496

6 Discussion

We should note that the improvement by using our method is just "on average," as in many other NLP tasks, and observing clear qualitative change is relatively difficult, for example, by just showing examples of similar word lists here. Comparing the results of BC_b and BC, Table 4 lists the numbers of improved, unchanged, and degraded words in terms of MP@20 for each evaluation set. As can be seen, there are a number of degraded words, although they are fewer than the improved words. Next, Figure 3 shows the averaged differences of MP@20 in each 40,000 word-ID range.[7] We can observe that the advantage of BC_b is lessened es-

[7] Word IDs are assigned in ascending order when we chose the top one million words as described in Section 5.2, and they roughly correlate with frequencies. So, frequent words tend to have low-IDs.

Table 4: The numbers of improved, unchanged, and degraded words in terms of MP@20 for each evaluation set.

	# improved	# unchanged	# degraded
Set A	755	2,585	400
Set B	643	2,610	404
Set C	3,153	3,962	1,738

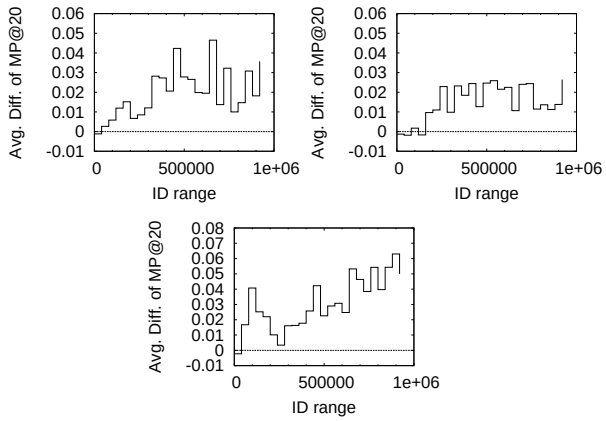


Figure 3: Averaged Differences of MP@20 between BC_b (0.0016) and BC within each 40,000 ID range (Left: Set A. Right: Set B. Bottom: Set C).

pecially for low-ID words (as expected) with on-average degradation.[8] The improvement is "on average" in this sense as well.

One might suspect that the answer words tended to be low-ID words, and the proposed method is simply biased towards low-ID words because of its nature. Then, the observed improvement is a trivial consequence. Table 5 lists some interesting statistics about the IDs. We can see that BC_b surely outputs more low-ID words than BC, and BC more than Cls-JS and JS.[9] However, the average ID of the outputs of BC is already lower than the average ID of the answer words. Therefore, even if BC_b preferred lower-ID words than BC, it should not have the effect of improving the accuracy. That is, the improvement by BC_b is not superficial. From BC/BC_b, we can also see that the IDs of the correct outputs did not become smaller compared to the IDs of the system outputs. Clearly, we need more analysis on what caused the improvement by the proposed method and how that affects the efficacy in real applications of similarity measures.

The proposed Bayesian similarity measure outperformed the baseline Bhattacharyya coefficient

[8]This suggests the use of different αs depending on ID ranges (e.g., smaller α for low-ID words) in practice.

[9]The outputs of Cls-JS are well-balanced in the ID space.

Table 5: Statistics on IDs. (A): Avg. ID of answers. (B): Avg. ID of system outputs. (C): Avg. ID of correct system outputs.

	Set A		Set C	
(A)	238,483		255,248	
	(B)	(C)	(B)	(C)
Cls-JS (s1+s2)	282,098	176,706	273,768	232,796
JS	183,054	11,3442	211,671	201,214
BC	162,758	98,433	193,508	189,345
$BC_b(0.0016)$	55,915	54,786	90,472	127,877
BC/BC_b	2.91	1.80	2.14	1.48

and other well-known similarity measures. As a smoothing method, it also outperformed a naive absolute discounting. Of course, we cannot say that the proposed method is better than any other sophisticated smoothing method at this point. However, as noted above, there has been no serious attempt to assess the effect of smoothing in the context of word similarity calculation. Recent studies have pointed out that the Bayesian framework derives state-of-the-art smoothing methods such as Kneser-Ney smoothing as a special case (Teh, 2006; Mochihashi et al., 2009). Consequently, it is reasonable to resort to the Bayesian framework. Conceptually, our method is equivalent to modifying $p(f_k|w_i)$ as $p(f_k|w_i) = \left\{\frac{\Gamma(\alpha_0+a_0)\Gamma(\alpha_k+c(w_i,f_k)+\frac{1}{2})}{\Gamma(\alpha_0+a_0+\frac{1}{2})\Gamma(\alpha_k+c(w_i,f_k))}\right\}^2$ and taking the Bhattacharyya coefficient. However, the implication of this form has not yet been investigated, and so we leave it for future research.

Our method is the simplest one as a Bayesian method. We did not employ any numerical optimization or sampling iterations, as in a more complete use of the Bayesian framework (Teh, 2006; Mochihashi et al., 2009). Instead, we used the obtained analytical form directly with the assumption that $\alpha_k = \alpha$ and α can be tuned directly by using a simple grid search with a small subset of the vocabulary as the development set. If substantial additional costs are allowed, we can fine-tune each α_k using more complete Bayesian methods. We also leave this for future research.

In terms of calculation procedure, BC_b has the same form as other similarity measures, which is basically the same as the inner product of sparse vectors. Thus, it can be as fast as other similarity measures with some effort as we described in Section 4 when our aim is to calculate similarities between words in a fixed large vocabulary. For example, BC_b took about 100 hours to calculate the

top 500 similar nouns for all of the one million nouns (using 16 CPU cores), while JS took about 57 hours. We think this is an acceptable additional cost.

The limitation of our method is that it cannot be used efficiently with similarity measures other than the Bhattacharyya coefficient, although that choice seems good as shown in the experiments. For example, it seems difficult to use the Jensen-Shannon divergence as the base similarity because the analytical form cannot be derived. One way we are considering to give more flexibility to our method is to adjust α_k depending on external knowledge such as the importance of a context (e.g., PMIs). In another direction, we will be able to use a "weighted" Bhattacharyya coefficient: $\sum_k \mu(w_1, f_k)\mu(w_2, f_k)\sqrt{p_{1k} \times p_{2k}}$, where the weights, $\mu(w_i, f_k)$, do not depend on p_{ik}, as the base similarity measure. The analytical form for it will be a weighted version of BC_b.

BC_b can also be generalized to the case where the base similarity is $BC^d(p_1, p_2) = \sum_{k=1}^{K} p_{1k}^d \times p_{2k}^d$, where $d > 0$. The Bayesian analytical form becomes as follows.

$$BC_b^d(w_1, w_2) = \frac{\Gamma(\alpha_0 + a_0)\Gamma(\beta_0 + b_0)}{\Gamma(\alpha_0 + a_0 + d)\Gamma(\beta_0 + b_0 + d)} \times \sum_{k=1}^{K} \frac{\Gamma(\alpha_k + c(w_1, f_k) + d)\Gamma(\beta_k + c(w_2, f_k) + d)}{\Gamma(\alpha_k + c(w_1, f_k))\Gamma(\beta_k + c(w_2, f_k))}.$$

See Appendix A for the derivation. However, we restricted ourselves to the case of $d = \frac{1}{2}$ in this study.

Finally, note that our BC_b is different from the Bhattacharyya distance measure on Dirichlet distributions of the following form described in Rauber et al. (2008) in its motivation and analytical form:

$$\frac{\sqrt{\Gamma(\alpha_0')\Gamma(\beta_0')}}{\sqrt{\prod_k \Gamma(\alpha_k')}\sqrt{\prod_k \Gamma(\beta_k')}} \times \frac{\prod_k \Gamma((\alpha_k' + \beta_k')/2)}{\Gamma(\frac{1}{2}\sum_k^K (\alpha_k' + \beta_k'))}. \quad (9)$$

Empirical and theoretical comparisons with this measure also form one of the future directions.[10]

7 Conclusion

We proposed a Bayesian method for robust distributional word similarities. Our method uses a distribution of context profiles obtained by Bayesian estimation and takes the expectation of a base similarity measure under that distribution. We showed that, in the case where the context profiles are multinomial distributions, the priors are Dirichlet, and the base measure is the Bhattacharyya coefficient, we can derive an analytical form, permitting efficient calculation. Experimental results show that the proposed measure gives better word similarities than a non-Bayesian Bhattacharyya coefficient, other well-known similarity measures such as Jensen-Shannon divergence and the cosine with PMI weights, and the Bhattacharyya coefficient with absolute discounting.

[10] Our preliminary experiments show that calculating similarity using Eq. 9 for the Dirichlet distributions obtained by Eq. 6 does not produce meaningful similarity (i.e., the accuracy is very low).

Appendix A

Here, we give the analytical form for the generalized case (BC_b^d) in Section 6. Recall the following relation, which is used to derive the normalization factor of the Dirichlet distribution:

$$\int_{\triangle} \prod_k \phi_k^{\alpha_k' - 1} d\phi = \frac{\prod_k \Gamma(\alpha_k')}{\Gamma(\alpha_0')} = Z(\alpha')^{-1}. \quad (10)$$

Then, $BC_b^d(w_1, w_2)$

$$= \iint_{\triangle \times \triangle} Dir(\phi_1|\alpha') Dir(\phi_2|\beta') \sum_k \phi_{1k}^d \phi_{2k}^d \, d\phi_1 \, d\phi_2$$
$$= Z(\alpha')Z(\beta') \times \underbrace{\iint_{\triangle \times \triangle} \prod_l \phi_{1l}^{\alpha_l' - 1} \prod_m \phi_{2m}^{\beta_m' - 1} \sum_k \phi_{1k}^d \phi_{2k}^d \, d\phi_1 \, d\phi_2}_{A}.$$

Using Eq. 10, A in the above can be calculated as follows:

$$= \int_{\triangle} \prod_m \phi_{2m}^{\beta_m' - 1} \left[\sum_k \phi_{2k}^d \int_{\triangle} \phi_{1k}^{\alpha_k' + d - 1} \prod_{l \neq k} \phi_{1l}^{\alpha_l' - 1} \, d\phi_1 \right] d\phi_2$$
$$= \int_{\triangle} \prod_m \phi_{2m}^{\beta_m' - 1} \left[\sum_k \phi_{2k}^d \frac{\Gamma(\alpha_k' + d) \prod_{l \neq k} \Gamma(\alpha_l')}{\Gamma(\alpha_0' + d)} \right] d\phi_2$$
$$= \sum_k \frac{\Gamma(\alpha_k' + d) \prod_{l \neq k} \Gamma(\alpha_l')}{\Gamma(\alpha_0' + d)} \int_{\triangle} \phi_{2k}^{\beta_k' + d - 1} \prod_{m \neq k} \phi_{2m}^{\beta_m' - 1} \, d\phi_2$$
$$= \sum_k \frac{\Gamma(\alpha_k' + d) \prod_{l \neq k} \Gamma(\alpha_l')}{\Gamma(\alpha_0' + d)} \frac{\Gamma(\beta_k' + d) \prod_{m \neq k} \Gamma(\beta_m')}{\Gamma(\beta_0' + d)}$$
$$= \frac{\prod \Gamma(\alpha_l') \prod \Gamma(\beta_m')}{\Gamma(\alpha_0' + d)\Gamma(\beta_0' + d)} \sum_k \frac{\Gamma(\alpha_k' + d)}{\Gamma(\alpha_k')} \frac{\Gamma(\beta_k' + d)}{\Gamma(\beta_k')}.$$

This will give:

$$BC_b^d(w_1, w_2) = \frac{\Gamma(\alpha_0')\Gamma(\beta_0')}{\Gamma(\alpha_0' + d)\Gamma(\beta_0' + d)} \sum_{k=1}^{K} \frac{\Gamma(\alpha_k' + d)\Gamma(\beta_k' + d)}{\Gamma(\alpha_k')\Gamma(\beta_k')}.$$

References

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 49:214–224.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. TR-10-98, Computer Science Group, Harvard University.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Corinna Cortes and Vladimir Vapnik. 1995. Support vector networks. *Machine Learning*, 20:273–297.

CRL. 2002. EDR electronic dictionary version 2.0 technical guide. Communications Research Laboratory (CRL).

Ido Dagan, Fernando Pereira, and Lillian Lee. 1994. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of ACL 94*.

Ido Dagan, Shaul Marcus, and Shaul Markovitch. 1995. Contextual word similarity and estimation from sparse data. *Computer, Speech and Language*, 9:123–152.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1997. Similarity-based methods for word sense disambiguation. In *Proceedings of ACL 97*.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Gregory Grefenstette. 1994. *Explorations In Automatic Thesaurus Discovery*. Kluwer Academic Publishers.

Zellig Harris. 1954. Distributional structure. *Word*, pages 146–142.

Donald Hindle. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*, pages 268–275.

Jun'ichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of ACL-08: HLT*.

Jun'ichi Kazama, Stijn De Saeger, Kentaro Torisawa, and Masaki Murata. 2009. Generating a large-scale analogy list using a probabilistic clustering based on noun-verb dependency profiles. In *Proceedings of 15th Annual Meeting of The Association for Natural Language Processing (in Japanese)*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pages 768–774.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of ACL-IJCNLP 2009*, pages 100–108.

Masaki Murata, Qing Ma, Tamotsu Shirado, and Hitoshi Isahara. 2004. Database for evaluating extracted terms and tool for visualizing the terms. In *Proceedings of LREC 2004 Workshop: Computational and Computer-Assisted Terminology*, pages 6–9.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP 2009*, pages 938–947.

T. W. Rauber, T. Braun, and K. Berns. 2008. Probabilistic distance measures of the Dirichlet and Beta distributions. *Pattern Recognition*, 41:637–645.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. Tsubaki: An open search engine infrastructure for developing new information access. In *Proceedings of IJCNLP 2008*.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of COLING-ACL 2006*, pages 985–992.

Akira Terada, Minoru Yoshida, and Hiroshi Nakagawa. 2004. A tool for constructing a synonym dictionary using context information. In *IPSJ SIG Technical Report (in Japanese)*, pages 87–94.

Recommendation in Internet Forums and Blogs

Jia Wang
Southwestern Univ. of Finance & Economics China
wj96@sina.cn

Qing Li
Southwestern Univ. of Finance & Economics China
liq_t@swufe.edu.cn

Yuanzhu Peter Chen
Memorial Univ. of Newfoundland Canada
yzchen@mun.ca

Zhangxi Lin
Texas Tech Univ. USA
zhangxi.lin@ttu.edu

Abstract

The variety of engaging interactions among users in social medial distinguishes it from traditional Web media. Such a feature should be utilized while attempting to provide intelligent services to social media participants. In this article, we present a framework to recommend relevant information in Internet forums and blogs using user comments, one of the most representative of user behaviors in online discussion. When incorporating user comments, we consider structural, semantic, and authority information carried by them. One of the most important observation from this work is that semantic contents of user comments can play a fairly different role in a different form of social media. When designing a recommendation system for this purpose, such a difference must be considered with caution.

1 Introduction

In the past twenty years, the Web has evolved from a framework of information dissemination to a social interaction facilitator for its users. From the initial dominance of static pages or sites, with addition of dynamic content generation and provision of client-side computation and event handling, Web applications have become a prevalent framework for distributed GUI applications. Such technological advancement has fertilized vibrant creation, sharing, and collaboration among the users (Ahn et al., 2007). As a result, the role of Computer Science is not as much of designing or implementing certain data communication techniques, but more of enabling a variety of creative uses of the Web.

In a more general context, Web is one of the most important carriers for "social media", e.g. Internet forums, blogs, wikis, podcasts, instant messaging, and social networking. Various engaging interactions among users in social media differentiate it from traditional Web sites. Such characteristics should be utilized in attempt to provide intelligent services to social media users. One form of such interactions of particular interest here is *user comments*. In *self-publication*, or *customer-generated media*, a user can publish an article or post news to share with others. Other users can read and comment on the posting and these comments can, in turn, be read and commented on. Digg (www.digg.com), Yahoo!Buzz (buzz.yahoo.com) and various kinds of blogs are commercial examples of self-publication. Therefore, reader responses to earlier discussion provide a valuable source of information for effective recommendation.

Currently, self-publishing media are becoming increasingly popular. For instance, at this point of writing, Technorati is indexing over 133 million blogs, and about 900,000 new blogs are created worldwide daily[1]. With such a large scale, information in the *blogosphere* follows a Long Tail Distribution (Agarwal et al., 2010). That is, in aggregate, the not-so-well-known blogs can have more valuable information than the popular ones. This gives us an incentive to develop a recommender to provide a set of relevant articles, which are expected to be of interest to the current reader. The user experience with the system can be immensely enhanced with the recommended articles. In this work, we focus on recommendation in Internet forums and blogs with discussion threads.

Here, a fundamental challenge is to account for topic divergence, i.e. the change of gist during the process of discussion. In a discussion thread, the original posting is typically followed by other readers' opinions, in the form of comments. Inten-

[1] http://technorati.com/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 257–265,
Uppsala, Sweden, 11-16 July 2010.

tion and concerns of active users may change as the discussion goes on. Therefore, recommendation, if it were only based on the original posting, can not benefit the potentially evolving interests of the users. Apparently, there is a need to consider topic evolution in adaptive content-based recommendation and this requires novel techniques in order to capture topic evolution precisely and to prevent drastic topic shifting which returns completely irrelevant articles to users.

In this work, we present a framework to recommend relevant information in Internet forums and blogs using user comments, one of the most representative recordings of user behaviors in these forms of social media.

It has the following contributions.

> The relevant information is recommended based on a balanced perspective of both the authors and readers.
>
> We model the relationship among comments and that relative to the original posting using graphs in order to evaluate their combined impact. In addition, the weight of a comment is further enhanced with its content and with the authority of its poster.

2 Related Work

In a broader context, a related problem is content-based information recommendation (or filtering). Most information recommender systems select articles based on the contents of the original postings. For instance, Chiang and Chen (Chiang and Chen, 2004) study a few classifiers for agent-based news recommendations. The relevant news selections of these work are determined by the textual similarity between the recommended news and the original news posting. A number of later proposals incorporate additional metadata, such as user behaviors and timestamps. For example, Claypool et al. (Claypool et al., 1999) combine the news content with numerical user ratings. Del Corso, Gullí, and Romani (Del Corso et al., 2005) use timestamps to favor more recent news. Cantador, Bellogin, and Castells (Cantador et al., 2008) utilize domain ontology. Lee and Park (Lee and Park, 2007) consider matching between news article attributes and user preferences. Anh et al. (Ahn et al., 2007) and Lai, Liang, and Ku (Lai et al., 2003) construct explicit user profiles, respectively. Lavrenko et al. (Lavrenko et al., 2000) propose the e-Analyst system which combines news stories with trends in financial time series. Some go even further by ignoring the news contents and only using browsing behaviors of the readers with similar interests (Das et al., 2007).

Another related problem is topic detection and tracking (TDT), i.e. automated categorization of news stories by their themes. TDT consists of breaking the stream of news into individual news stories, monitoring the stories for events that have not been seen before, and categorizing them (Lavrenko and Croft, 2001). A topic is modeled with a language profile deduced by the news. Most existing TDT schemes calculate the similarity between a piece of news and a topic profile to determine its topic relevance (Lavrenko and Croft, 2001) (Yang et al., 1999). Qiu (Qiu et al., 2009) apply TDT techniques to group news for collaborative news recommendation. Some work on TDT takes one step further in that they update the topic profiles as part of the learning process during its operation (Allan et al., 2002) (Leek et al., 2002).

Most recent researches on information recommendation in social media focus on the blogosphere. Various types of user interactions in the blogosphere have been observed. A prominent feature of the blogosphere is the *collective wisdom* (Agarwal et al., 2010). That is, the knowledge in the blogosphere is enriched by such engaging interactions among bloggers and readers as posting, commenting and tagging. Prior to this work, the linking structure and user tagging mechanisms in the blogosphere are the most widely adopted ones to model such collective wisdom. For example, Esmaili et al. (Esmaili et al., 2006) focus on the linking structure among blogs. Hayes, Avesani, and Bojars (Hayes et al., 2007) explore measures based on blog authorship and reader tagging to improve recommendation. Li and Chen further integrate trust, social relation and semantic analysis (Li and Chen, 2009). These approaches attempt to capture accurate similarities between postings without using reader comments. Due to the interactions between bloggers and readers, blog recommendation should not limit its input to only blog postings themselves but also incorporate feedbacks from the readers.

The rest of this article is organized as follows. We first describe the design of our recommendation framework in Section 3. We then evaluate the performance of such a recommender using two

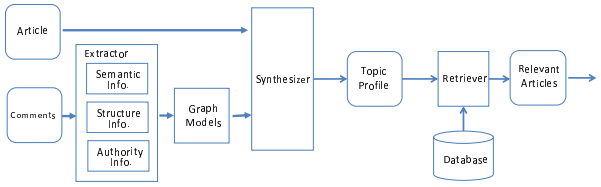


Figure 1: Design scheme

different social media corpora (Section 4). This paper is concluded with speculation on how the current prototype can be further improved in Section 5.

3 System Design

In this section, we present a mechanism for recommendation in Internet forums and blogs. The framework is sketched in Figure 1. Essentially, it builds a topic profile for each original posting along with the comments from readers, and uses this profile to retrieve relevant articles. In particular, we first extract structural, semantic, and authority information carried by the comments. Then, with such collective wisdom, we use a graph to model the relationship among comments and that relative to the original posting in order to evaluate the impact of each comment. The graph is weighted with postings' contents and the authors' authority. This information along with the original posting and its comments are fed into a synthesizer. The synthesizer balances views from both authors and readers to construct a topic profile to retrieve relevant articles.

3.1 Incorporating Comments

In a discussion thread, comments made at different levels reflect the variation of focus of readers. Therefore, recommended articles should reflect their concerns to complement the author's opinion. The degree of contribution from each comment, however, is different. In the extreme case, some of them are even advertisements which are completely irrelevant to the discussion topics. In this work, we use a graph model to differentiate the importance of each comment. That is, we model the authority, semantic, structural relations of comments to determine their combined impact.

3.1.1 Authority Scoring Comments

Intuitively, each comment may have a different degree of authority determined by the status of its author (Hu et al., 2007). Assume we have users in a forum, denoted by . We calculate the authority for user . To do that, we employ a variant of the PageRank algorithm (Brin and Page, 1998). We consider the cases that a user replies to a previous posting and that a user quotes a previous posting separately. For user , we use to denote the number of times that has replied to user . Similarly, we use to denote the number of times that has quoted user . We combine them linearly:

Further, we normalize the above quantity to record how frequently a user refers to another:

Inline with the PageRank algorithm, we define the authority of user as

–

3.1.2 Differentiating comments with Semantic and Structural relations

Next, we construct a similar model in terms of the comments themselves. In this model, we treat the original posting and the comments each as a text node. This model considers both the content similarity between text nodes and the logic relationship among them.

On the one hand, the semantic similarity between two nodes can be measured with any commonly adopted metric, such as cosine similarity and Jaccard coefficient (Baeza-Yates and Ribeiro-Neto, 1999). On the other hand, the structural relation between a pair of nodes takes two forms as we have discussed earlier. First, a comment can be made in response to the original posting or at most one earlier comment. In graph theoretic terms, the hierarchy can be represented as a tree , where is the set of all text nodes and is the edge set. In particular, the original posting is the root and all the comments are ordinary nodes. There is an arc (directed edge) from node to node , denoted , if the corresponding comment is made in response to comment (or original posting) . Second, a comment can quote from one or more earlier comments. From this perspective, the hierarchy can be modeled using a directed acyclic graph (DAG),

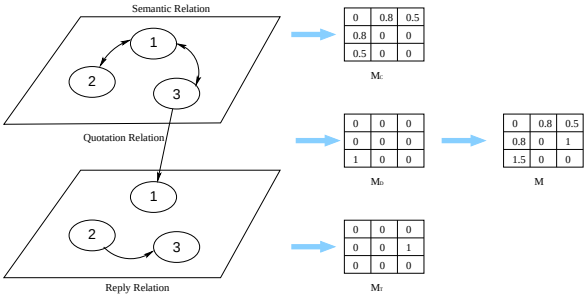


Figure 2: Multi-relation graph of comments based on the structural and semantic information

denoted . There is an arc from node to node , denoted , if the corresponding comment quotes comment (or original posting) . As shown in Figure 2, for either graph or , we can use a adjacency matrix, denoted and , respectively, to record them. Similarly, we can also use a matrix defined on to record the content similarity between nodes and denote it by . Thus, we combine these three aspects linearly:

The importance of a text node can be quantized by the times it has been referred to. Considering the semantic similarity between nodes, we use another variant of the PageRank algorithm to calculate the weight of comment :

—

where is a damping factor, and is the normalized weight of comment referring to defined as

———————

where is an entry in the graph adjacency matrix M and is a constant to avoid division by zero.

In some social networking media, a user may have a subset of other users as "friends". This can be captured by a matrix of , whose entries are denoted by . Thus, with this information and assuming poster has made a comment k for user 's posting, the final weight of this comment is defined as

————————

3.2 Topic Profile Construction

Once the weight of comments on one posting is quantified by our models, this information along with the entire discussion thread is fed into a synthesizer to construct a topic profile. As such, the perspectives of both authors and readers are balanced for recommendation.

The profile is a weight vector of terms to model the language used in the discussion thread. Consider a posting and its comment sequence . For each term , a compound weight is calculated. It is a linear combination of the contribution by the posting itself, , and that by the comments, . We assume that each term is associated with an "inverted document frequency", denoted by —, where is the corpus size and is the number of documents in corpus containing term . We use a function to denote the number of occurrences of term in document , i.e. "term frequency". Thus, when the original posting and comments are each considered as a document, this term frequency can be calculated for any term in any document. We thus define the weight of term in document , be the posting itself or a comment, using the standard TF/IDF definition (Baeza-Yates and Ribeiro-Neto, 1999):

———————

The weight contributed by the posting itself, , is thus:

———————

The weight contribution from the comments incorporates not only the language features of these documents but also their importance in the discussion thread. That is, the contribution of comment score is incorporated into weight calculation of the words in a comment.

——————— ————————

Such a treatment of compounded weight is essentially to recognize that readers' impact on selecting relevant articles and the difference of their influence. For each profile, we select the top- highest weighted words to represent the topic.

With the topic profile thus constructed, the retriever returns an ordered list of articles with decreasing relevance to the topic. Note that our approach to differentiate the importance of each comment can be easily incorporated into any generic retrieval model. In this work, our retriever is adopted from (Lavrenko et al., 2000).

3.3 Interpretation of Recommendation

Since interpreting recommended items enhances users' trusting beliefs (Wang and Benbasat, 2007), we design a creative approach to generate hints to indicate the relationship (generalization, specialization and duplication) between the recommended articles and the original posting based on our previous work (Candan et al., 2009).

Article being more general than can be interpreted as being less constrained than by the keywords they contain. Let us consider two articles, and , where contains keywords, and , and only contains .

> If is said to be more general than , then the additional keyword, , of article must render less constrained than . Therefore, the content of can be interpreted as .
>
> If, on the other hand, is said to be more specific than , then the additional keyword, , must render more constrained than . Therefore, the content of can be interpreted as .

Note that, in the two-keyword space , can be denoted by a vector and can be denoted by . The origin corresponds to the case where an article does contain neither nor . That is, corresponds to an article which can be interpreted as . Therefore, if is said to be more general than , should be greater than . This allows us to measure the degrees of generalization and specialization of two articles. Given two articles, and , *of the same topic*, they will have a common keyword base, while both articles will also have their own content, different from their common base. Let us denote the common part of by and common part of by . Note that and are usually unequal because the same words in the common part have different term weights in article and respectively. Given these and the generalization concept introduced above for two similar articles and , we can define the degree of generalization ( ) and specialization ( ) of with respect to as

To alleviate the effect of document length, we revise the definition as

The relative specialization and generalization values can be used to reveal the relationships between recommended articles and the original posting. Given original posting and recommended article , if , for a given generalization threshold , then B is marked as a generalization. When this is not the case, if , for a given specialization threshold, , then is marked as a specialization. If neither of these cases is true, then is duplicate of .

Such an interpretation provides a control on delivering recommended articles. In particular, we can filter the duplicate articles to avoid recommending the same information.

4 Experimental Evaluation

To evaluate the effectiveness of our proposed recommendation mechanism, we carry out a series of experiments on two synthetic data sets, collected from Internet forums and blogs, respectively. The first data set is called Forum. This data set is constructed by randomly selecting 20 news articles with corresponding reader comments from the Digg Web site and 16,718 news articles from the Reuters news Web site. This simulates the scenario of recommending relevant news from traditional media to social media users for their further reading. The second one is the Blog data set containing 15 blog articles with user comments and 15,110 articles obtained from the Myhome Web site [2]. Details of these two data sets are shown in Table 1. For evaluation purposes, we adopt the traditional pooling strategy (Zobel, 1998) and apply to the TREC data set to mark the relevant articles for each topic.

[2] http://blogs.myhome.ie

Table 1: Evaluation data set

Synthetic Data Set		Forum	Blog
Topics	No. of postings	20	15
	Ave. length of postings	676	236
	No. of comments per posting	81.4	46
	Ave. length of comments	45	150
Target	No. of articles	16718	15110
	Ave. length of articles	583	317

The recommendation engine may return a set of essentially the same articles re-posted at different sites. Therefore, we introduce a metric of novelty to measure the topic diversity of returned suggestions. In our experiments, we define precision and novelty metrics as

——— and ———

where is the subset of the top- articles returned by the recommender, is the set of manually tagged relevant articles, and is the set of manually tagged relevant articles excluding duplicate ones to the original posting. We select the top 10 articles for evaluation assuming most readers only browse up to 10 recommended articles (Karypis, 2001). Meanwhile, we also utilize mean average precision (MAP) and mean average novelty (MAN) to evaluate the entire set of returned article.

We test our proposal in four aspects. First, we compare our work to two baseline works. We then present results for some preliminary tests to find out the optimal values for two critical parameters. Next, we study the effect of user authority and its integration to comment weighting. Fourth, we evaluate the performance gain obtained from interpreting recommendation. In addition, we provide a significance test to show that the observed differences in effectiveness for different approaches are not incidental. In particular, we use the -test here, which is commonly used for significance tests in information retrieval experiments (Hull, 1993).

4.1 Overall Performance

As baseline proposals, we also implement two well-known content-based recommendation methods (Bogers and Bosch, 2007). The first method, Okapi, is commonly applied as a representative of the classic probabilistic model for relevant information retrieval (Robertson and Walker, 1994). The second one, LM, is based on statistical language models for relevant information retrieval (Ponte and Croft, 1998). It builds a probabilistic language model for each article, and ranks them on query likelihood, i.e. the probability of the model generating the query. Following the strategy of Bogers and Bosch, relevant articles are selected based on the title and the first 10 sentences of the original postings. This is because articles are organized in the so-called *inverted pyramid* style, meaning that the most important information is usually placed at the beginning. Trimming the rest of an article would usually remove relatively less crucial information, which speeds up the recommendation process.

Table 2: Overall performance

Data	Method	Precision		Novelty	
Forum	**Okapi**	0.827	0.833	0.807	0.751
	LM	0.804	0.833	0.807	0.731
	Our	0.967	0.967	0.9	0.85
Blog	**Okapi**	0.733	0.651	0.667	0.466
	LM	0.767	0.718	0.70	0.524
	Our	0.933	0.894	0.867	0.756

A paired -test shows that using and as performance measures, our approach performs significantly better than the baseline methods for both Forum and Blog data sets as shown in Table 2. In addition, we conduct -tests using MAP and MAN as performance measures, respectively, and the -values of these tests are all less than 0.05, meaning that the results of experiments are statistically significant. We believe that such gains are introduced by the additional information from the collective wisdom, i.e. user authority and comments. Note that the retrieval precision for Blog of two baseline methods is not as good as that for Forum. Our explanation is that blog articles may not be organized in the inverted pyramid style as strictly as news forum articles.

4.2 Parameters of Topic Profile

There are two important parameters to be considered to construct topic profiles for recommendation. 1) the number of the most weighted words to represent the topic, and 2) combination coefficient to determine the contribution of original posting and comments in selecting relevant articles.We conduct a series of experiments and find out that the optimal performance is obtained when the number of words is between 50 and 70, and is between 0.65 and 0.75. When is set to 0, the recommended articles only reflect the author's opinion. When , the suggested articles represent the concerns of readers exclusively. In the

Table 3: Performance of four runs

	Method	Precision		Novelty	
Forum	RUN1	0.88	0.869	0.853	0.794
	RUN2	0.933	0.911	0.9	0.814
	RUN3	0.94	0.932	0.9	0.848
	RUN4	0.967	0.967	0.9	0.85
Blog	RUN1	0.767	0.758	0.7	0.574
	RUN2	0.867	0.828	0.833	0.739
	RUN3	0.9	0.858	0.833	0.728
	RUN4	0.933	0.894	0.867	0.756

following experiments, we set topic word number to 60 and combination coefficient to 0.7.

4.3 Effect of Authority and Comments

In this part, we explore the contribution of user authority and comments in social media recommender. In particular, we study the following scenarios with increasing system capabilities. Note that, lacking friend information (Section 3.1.2) in the Forum data set, is set to zero.

RUN 1 (Posting): the topic profile is constructed only based on the original posting itself. This is analogous to traditional recommenders which only consider the focus of authors for suggesting further readings.

RUN 2 (Posting+Authority): the topic profile is constructed based on the original posting and participant authority.

RUN 3 (Posting+Comment): the topic profile is constructed based on the original posting and its comments.

RUN 4 (All): the topic profile is constructed based on the original posting, user authority, and its comments.

Here, we set . Our -test shows that using and as performance measures, RUN4 performs best in both Forum and Blog data sets as shown in Table 3. There is a stepwise performance improvement while integrating user authority, comments and both. With the assistance of user authority and comments, the recommendation precision is improved up to 9.8% and 21.6% for Forum and Blog, respectively. The opinion of readers is an effective complementarity to the authors' view in suggesting relevant information for further reading.

Moreover, we investigate the effect of the semantic and structural relations among comments, i.e. semantic similarity, reply, and quotation. For this purpose, we carry out a series of experiments based on different combinations of these relations.

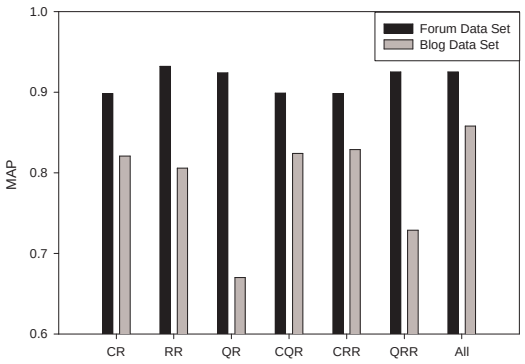


Figure 3: Effect of content, quotation and reply relation

Content Relation (CR): only the content relation matrix is used in scoring the comments.

Quotation Relation (QR): only the quotation relation matrix is used in scoring the comments.

Reply Relation (RR): only the reply relation matrix is used in scoring the comments.

Content+Quotation Relation (CQR): both the content and quotation relation matrices is used in scoring the comments.

Content+Reply Relation(CRR): both the content and reply relation matrices are used in scoring the comments.

Quotation+Reply Relation (QRR): both the quotation and reply relation matrices are used in scoring the comments.

All: all three matrices are used.

The MAP yielded by these combinations for both data sets is plotted in Figure 3. For the case of Forum, we observe that incorporating content information adversely affects recommendation precision. This concurs with what we saw in our previous work (Wang et al., 2010). On the other hand, when we test the Blog data set, the trend is the opposite, i.e. content similarity does contribute to retrieval performance positively. This is attributed by the text characteristics of these two forms of social media. Specifically, comments in news forums usually carry much richer structural information than blogs where comments are usually "flat" among themselves.

4.4 Recommendation Interpretation

To evaluate the precision of interpreting the relationship between recommended articles and the

original posting, the evaluation metric of success rate is defined as

where is the number of recommended articles, is the error weight of recommended article . Here, the error weight is set to one if the result interpretation is mis-labelled.

From our studies, we observe that the success rate at top-10 is around 89.3% and 87.5% for the Forum and Blog data sets, respectively. Note that these rates include the errors introduced by the irrelevant articles returned by the retrieval module. To estimate optimal thresholds of generalization and specialization, we calculate the success rate at different threshold values and find that neither too small nor too large a value is appropriate for interpretation. In our experiments, we set generalization threshold to 3.2 and specialization threshold to 1.8 for the Forum data set, and to 3.5 and to 2.0 for Blog. Ideally, threshold values would need to be set through a machine learning process, which identifies proper values based on a given training sample.

5 Conclusion and Future Work

The Web has become a platform for social networking, in addition to information dissemination at its earlier stage. Many of its applications are also being extended in this fashion. Traditional recommendation is essentially a push service to provide information according to the profile of individual or groups of users. Its niche at the Web 2.0 era lies in its ability to enable online discussion by serving up relevant references to the participants. In this work, we present a framework for information recommendation in such social media as Internet forums and blogs. This model incorporates information of user status and comment semantics and structures within the entire discussion thread. This framework models the logic connections among readers and the innovativeness of comments. By combining such information with traditional statistical language models, it is capable of suggesting relevant articles that meet the dynamic nature of a discussion in social media. One important discovery from this work is that, when integrating comment contents, the structural information among comments, and reader relationship, it is crucial to distinguish the characteristics of various forms of social media. The reason is that the role that the semantic content of a comment plays can differ from one form to another.

This study can be extended in a few interesting ways. For example, we can also evaluate its effectiveness and costs during the operation of a discussion forum, where the discussion thread is continually updated by new comments and votes. Indeed, its power is yet to be further improved and investigated.

Acknowledgments

Li's research is supported by National Natural Science Foundation of China (Grant No.60803106), the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, and the Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China. Research of Chen is supported by Natural Science and Engineering Council (NSERC) of Canada.

References

Nitin Agarwal, Magdiel Galan, Huan Liu, and Shankar Subramanya. 2010. Wiscoll: Collective wisdom based blog clustering. *Information Sciences*, 180(1):39–61.

Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. 2007. Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 11–20.

James Allan, Victor Lavrenko, and Russell Swan. 2002. Explorations within topic tracking and detection. Topic detection and tracking: event-based information organization *Kluwer Academic Publishers*, pages 197–224.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. Modern information retrieval. *Addison Wesley Longman Publisher*.

Toine Bogers and Antal Bosch. 2007. Comparing and evaluating information retrieval algorithms for news recommendation. In *Proceedings of 2007 ACM conference on Recommender Systems*, pages 141–144.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

K. Selçuk Candan, Mehmet E. Dönderler, Terri Hedgpeth, Jong Wook Kim, Qing Li, and Maria Luisa Sapino. 2009. SEA: Segment-enrich-annotate paradigm for adapting dialog-based content for improved accessibility. *ACM Transactions on Information Systems (TOIS)*, 27(3):1–45.

Ivan Cantador, Alejandro Bellogin, and Pablo Castells. 2008. Ontology-based personalized and context-aware recommendations of news items. In *Proceedings of IEEE/WIC/ACM international Conference on Web Intelligence and Intelligent Agent Technology (WI)*, pages 562–565.

Jung-Hsien Chiang and Yan-Cheng Chen. 2004. An intelligent news recommender agent for filtering and categorizing large volumes of text corpus. *International Journal of Intelligent Systems*, 19(3):201–216.

Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. 1999. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 271–280.

Gianna M. Del Corso, Antonio Gullí, and Francesco Romani. 2005. Ranking a stream of news. In *Proceedings of the 14th International Conference on World Wide Web(WWW)*, pages 97–106.

Kyumars Sheykh Esmaili, Mahmood Neshati, Mohsen Jamali, Hassan Abolhassani, and Jafar Habibi. 2006. Comparing performance of recommendation techniques in the blogsphere. In *ECAI 2006 Workshop on Recommender Systems*.

Conor Hayes, Paolo Avesani, and Uldis Bojars. 2007. An analysis of bloggers, topics and tags for a blog recommender system. In *Workshop on Web Mining (WebMine)*, pages 1–20.

Meishan Hu, Aixin Sun, and Ee-Peng Lim. 2007. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the sixteenth ACM Conference on Conference on Information and Knowledge Management(CIKM)*, pages 901–904.

David Hull. 1993. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338.

George Karypis. 2001. Evaluation of item-based Top-N recommendation algorithms. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)*, pages 247–254.

Hung-Jen Lai, Ting-Peng Liang, and Yi Cheng Ku. 2003. Customized internet news services based on customer profiles. In *Proceedings of the 5th International Conference on Electronic commerce (ICEC)*, pages 225–229.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Language models for financial news recommendation. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*, pages 389–396.

Hong Joo Lee and Sung Joo Park. 2007. MONERS: A news recommender for the mobile web. *Expert Systems with Applications*, 32(1):143–150.

Tim Leek, Richard Schwartz, and Srinivasa Sista. 2002. Probabilistic approaches to topic detection and tracking. *Topic detection and tracking: event-based information organization*, pages 67–83.

Yung-Ming Li and Ching-Wen Chen. 2009. A synthetical approach for blog recommendation: Combining trust, social relation, and semantic analysis. *Expert Systems with Applications*, 36(3):6536 – 6547.

Jay Michael Ponte and William Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281.

Jing Qiu, Lejian Liao, and Peng Li. 2009. News recommender system based on topic detection and tracking. In *Proceedings of the 4th Rough Sets and Knowledge Technology*.

Stephen E. Robertson and Stephen G Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM SIGIR conference on Research and Development in Information Retrieval*, pages 232–241.

Weiquan Wang and Izak Benbasat. 2007. Recommendation agents for electronic commerce: Effects of explanation facilities on trusting beliefs. *Journal of Management Information Systems*, 23(4):217–246.

Jia Wang, Qing Li, and Yuanzhu Peter Chen. 2010. User comments for news recommendation in social media. In *Proceedings of the 33rd ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 295–296.

Yiming Yang, Jaime Guillermo Carbonell, Ralf D. Brown, Thomas Pierce, Brian T. Archibald, and Xin Liu. 1999. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43.

Justin Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314.

Learning Phrase-Based Spelling Error Models from Clickthrough Data

Xu Sun*
Dept. of Mathematical Informatics
University of Tokyo, Tokyo, Japan
xusun@mist.i.u-tokyo.ac.jp

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Daniel Micol
Microsoft Corporation
Munich, Germany
danielmi@microsoft.com

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

Abstract

This paper explores the use of clickthrough data for query spelling correction. First, large amounts of query-correction pairs are derived by analyzing users' query reformulation behavior encoded in the clickthrough data. Then, a phrase-based error model that accounts for the transformation probability between multi-term phrases is trained and integrated into a query speller system. Experiments are carried out on a human-labeled data set. Results show that the system using the phrase-based error model outperforms significantly its baseline systems.

1 Introduction

Search queries present a particular challenge for traditional spelling correction methods for three main reasons (Ahmad and Kondrak, 2004). First, spelling errors are more common in search queries than in regular written text: roughly 10-15% of queries contain misspelled terms (Cucerzan and Brill, 2004). Second, most search queries consist of a few key words rather than grammatical sentences, making a grammar-based approach inappropriate. Most importantly, many queries contain search terms, such as proper nouns and names, which are not well established in the language. For example, Chen et al. (2007) reported that 16.5% of valid search terms do not occur in their 200K-entry spelling lexicon.

Therefore, recent research has focused on the use of Web corpora and query logs, rather than human-compiled lexicons, to infer knowledge about misspellings and word usage in search queries (e.g., Whitelaw et al., 2009). Another important data source that would be useful for this purpose is clickthrough data. Although it is well-known that clickthrough data contain rich information about users' search behavior, e.g., how a user (re-) formulates a query in order to find the relevant document, there has been little research on exploiting the data for the development of a query speller system.

In this paper we present a novel method of extracting large amounts of query-correction pairs from the clickthrough data. These pairs, implicitly judged by millions of users, are used to train a set of spelling error models. Among these models, the most effective one is a phrase-based error model that captures the probability of transforming one multi-term phrase into another multi-term phrase. Comparing to traditional error models that account for transformation probabilities between single characters (Kernighan et al., 1990) or sub-word strings (Brill and Moore, 2000), the phrase-based model is more powerful in that it captures some contextual information by retaining inter-term dependencies. We show that this information is crucial to detect the correction of a query term, because unlike in regular written text, any query word can be a valid search term and in many cases the only way for a speller system to make the judgment is to explore its usage according to the contextual information.

We conduct a set of experiments on a large data set, consisting of human-labeled

* The work was done when Xu Sun was visiting Microsoft Research Redmond.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 266–274,
Uppsala, Sweden, 11-16 July 2010.

query-correction pairs. Results show that the error models learned from clickthrough data lead to significant improvements on the task of query spelling correction. In particular, the speller system incorporating a phrase-based error model significantly outperforms its baseline systems.

To the best of our knowledge, this is the first extensive study of learning phase-based error models from clickthrough data for query spelling correction. The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 presents the way query-correction pairs are extracted from the clickthrough data. Section 4 presents the baseline speller system used in this study. Section 5 describes in detail the phrase-based error model. Section 6 presents the experiments. Section 7 concludes the paper.

2 Related Work

Spelling correction for regular written text is a long standing research topic. Previous researches can be roughly grouped into two categories: correcting non-word errors and real-word errors.

In non-word error spelling correction, any word that is not found in a pre-compiled lexicon is considered to be misspelled. Then, a list of lexical words that are *similar* to the misspelled word are proposed as candidate spelling corrections. Most traditional systems use a manually tuned similarity function (e.g., edit distance function) to rank the candidates, as reviewed by Kukich (1992). During the last two decades, statistical error models learned on training data (i.e., query-correction pairs) have become increasingly popular, and have proven more effective (Kernighan et al., 1990; Brill and Moore, 2000; Toutanova and Moore, 2002; Okazaki et al., 2008).

Real-word spelling correction is also referred to as context sensitive spelling correction (CSSC). It tries to detect incorrect usages of a valid word based on its context, such as "peace" and "piece" in the context "a _ of cake". A common strategy in CSSC is as follows. First, a pre-defined confusion set is used to generate candidate corrections, then a scoring model, such as a trigram language model or naïve Bayes classifier, is used to rank the candidates according to their context (e.g., Golding and Roth, 1996; Mangu and Brill, 1997; Church et al., 2007).

When designed to handle regular written text, both CSSC and non-word error speller systems rely on a pre-defined vocabulary (i.e., either a lexicon or a confusion set). However, in query spelling correction, it is impossible to compile such a vocabulary, and the boundary between the non-word and real-word errors is quite vague. Therefore, recent research on query spelling correction has focused on exploiting noisy Web data and query logs to infer knowledge about misspellings and word usage in search queries. Cucerzan and Brill (2004) discuss in detail the challenges of query spelling correction, and suggest the use of query logs. Ahmad and Kondrak (2005) propose a method of estimating an error model from query logs using the EM algorithm. Li et al. (2006) extend the error model by capturing word-level similarities learned from query logs. Chen et al. (2007) suggest using web search results to improve spelling correction. Whitelaw et al. (2009) present a query speller system in which both the error model and the language model are trained using Web data.

Compared to Web corpora and query logs, clickthrough data contain much richer information about users' search behavior. Although there has been a lot of research on using clickthrough data to improve Web document retrieval (e.g., Joachims, 2002; Agichtein et al., 2006; Gao et al., 2009), the data have not been fully explored for query spelling correction. This study tries to learn error models from clickthrough data. To our knowledge, this is the first such attempt using clickthrough data.

Most of the speller systems reviewed above are based on the framework of the source channel model. Typically, a language model (source model) is used to capture contextual information, while an error model (channel model) is considered to be *context free* in that it does not take into account any contextual information in modeling word transformation probabilities. In this study we argue that it is beneficial to capture contextual information in the error model. To this end, inspired by the phrase-based statistical machine translation (SMT) systems (Koehn et al., 2003; Och and Ney, 2004), we propose a phrase-based error model where we assume that query spelling correction is performed at the phrase level.

In what follows, before presenting the phrase-based error model, we will first describe the clickthrough data and the query speller system we used in this study.

3 Clickthrough Data and Spelling Correction

This section describes the way the query-correction pairs are extracted from click-

through data. Two types of clickthrough data are explored in our experiment.

The clickthrough data of the first type has been widely used in previous research and proved to be useful for Web search (Joachims, 2002; Agichtein et al., 2006; Gao et al., 2009) and query reformulation (Wang and Zhai, 2008; Suzuki et al., 2009). We start with this same data with the hope of achieving similar improvements in our task. The data consist of a set of query sessions that were extracted from one year of log files from a commercial Web search engine. A query session contains a query issued by a user and a ranked list of links (i.e., URLs) returned to that same user along with records of which URLs were clicked. Following Suzuki et al. (2009), we extract query-correction pairs as follows. First, we extract pairs of queries Q_1 and Q_2 such that (1) they are issued by the same user; (2) Q_2 was issued within 3 minutes of Q_1; and (3) Q_2 contained at least one clicked URL in the result page while Q_1 did not result in any clicks. We then scored each query pair (Q_1, Q_2) using the edit distance between Q_1 and Q_2, and retained those with an edit distance score lower than a pre-set threshold as query correction pairs.

Unfortunately, we found in our experiments that the pairs extracted using the method are too noisy for reliable error model training, even with a very tight threshold, and we did not see any significant improvement. Therefore, in Section 6 we will not report results using this dataset.

The clickthrough data of the second type consists of a set of *query reformulation sessions* extracted from 3 months of log files from a commercial Web browser. A query reformulation session contains a list of URLs that record user behaviors that relate to the query reformulation functions, provided by a Web search engine. For example, almost all commercial search engines offer the "did you mean" function, suggesting a possible alternate interpretation or spelling of a user-issued query. Figure 1 shows a sample of the query reformulation sessions that record the "did you mean" sessions from three of the most popular search engines. These sessions encode the same user behavior: A user first queries for "harrypotter sheme park", and then clicks on the resulting spelling suggestion "harry potter theme park". In our experiments, we "reverse-engineer" the parameters from the URLs of these sessions, and deduce how each search engine encodes both a query and the fact that a user arrived at a URL by clicking on the spelling suggestion of the query – an important indication that the spelling suggestion is desired. From these three months of query reformulation sessions, we extracted about 3 million query-correction pairs. Compared to the pairs extracted from the clickthrough data of the first type (query sessions), this data set is much cleaner because all these spelling corrections are actually clicked, and thus judged implicitly, by many users.

Google:

```
http://www.google.com/search?
hl=en&source=hp&
q=harrypotter+sheme+park&aq=f&oq=&aqi=

http://www.google.com/search?
hl=en&ei=rnNAS8-oKsWe_AaB2eHlCA&
sa=X&oi=spell&resnum=0&ct=
result&cd=1&ved=0CA4QBSgA&
q=harry+potter+theme+park&spell=1
```

Yahoo:

```
http://search.yahoo.com/search;
_ylt=A0geu6ywckBL_XIBSDtXNyoA?
p=harrypotter+sheme+park&
fr2=sb-top&fr=yfp-t-701&sao=1

http://search.yahoo.com/search?
ei=UTF-8&fr=yfp-t-701&
p=harry+potter+theme+park
&SpellState=n-2672070758_q-tsI55N6srhZa.
qORA0MuawAAAA%40%40&fr2=sp-top
```

Bing:

```
http://www.bing.com/search?
q=harrypotter+sheme+park&form=QBRE&qs=n

http://www.bing.com/search?
q=harry+potter+theme+park&FORM=SSRE
```

Figure 1. A sample of query reformulation sessions from three popular search engines. These sessions show that a user first issues the query "harrypotter sheme park", and then clicks on the resulting spell suggestion "harry potter theme park".

In addition to the "did you mean" function, recently some search engines have introduced two new spelling suggestion functions. One is the "auto-correction" function, where the search engine is confident enough to automatically apply the spelling correction to the query and execute it to produce search results for the user. The other is the "split pane" result page, where one half portion of the search results are produced using the original query, while the other half, usually visually separate portion of results are produced using the auto-corrected query.

In neither of these functions does the user ever receive an opportunity to approve or disapprove of the correction. Since our extraction approach focuses on user-approved spelling suggestions,

we ignore the query reformulation sessions recording either of the two functions. Although by doing so we could miss some basic, obvious spelling corrections, our experiments show that the negative impact on error model training is negligible. One possible reason is that our baseline system, which does not use any error model learned from the clickthrough data, is already able to correct these basic, obvious spelling mistakes. Thus, including these data for training is unlikely to bring any further improvement.

We found that the error models trained using the data directly extracted from the query reformulation sessions suffer from the problem of underestimating the self-transformation probability of a query $P(Q_2=Q_1|Q_1)$, because we only included in the training data the pairs where the query is different from the correction. To deal with this problem, we augmented the training data by including correctly spelled queries, i.e., the pairs (Q_1, Q_2) where $Q_1 = Q_2$. First, we extracted a set of queries from the sessions where no spell suggestion is presented or clicked on. Second, we removed from the set those queries that were recognized as being auto-corrected by a search engine. We do so by running a sanity check of the queries against our baseline spelling correction system, which will be described in Section 6. If the system thinks an input query is misspelled, we assumed it was an obvious misspelling, and removed it. The remaining queries were assumed to be correctly spelled and were added to the training data.

4 The Baseline Speller System

The spelling correction problem is typically formulated under the framework of the source channel model. Given an input query $Q = q_1 \dots q_I$, we want to find the best spelling correction $C = c_1 \dots c_J$ among all candidate spelling corrections:

$$C^* = \underset{C}{\operatorname{argmax}}\, P(C|Q) \quad (1)$$

Applying Bayes' Rule and dropping the constant denominator, we have

$$C^* = \underset{C}{\operatorname{argmax}}\, P(Q|C)P(C) \quad (2)$$

where the error model $P(Q|C)$ models the transformation probability from C to Q, and the language model $P(C)$ models how likely C is a correctly spelled query.

The speller system used in our experiments is based on a ranking model (or ranker), which can be viewed as a generalization of the source channel model. The system consists of two components: (1) a candidate generator, and (2) a ranker.

In candidate generation, an input query is first tokenized into a sequence of terms. Then we scan the query from left to right, and each query term q is looked up in lexicon to generate a list of spelling suggestions c whose edit distance from q is lower than a preset threshold. The lexicon we used contains around 430,000 entries; these are high frequency query terms collected from one year of search query logs. The lexicon is stored using a trie-based data structure that allows efficient search for all terms within a maximum edit distance.

The set of all the generated spelling suggestions is stored using a lattice data structure, which is a compact representation of exponentially many possible candidate spelling corrections. We then use a decoder to identify the top twenty candidates from the lattice according to the source channel model of Equation (2). The language model (the second factor) is a backoff bigram model trained on the tokenized form of one year of query logs, using maximum likelihood estimation with absolute discounting smoothing. The error model (the first factor) is approximated by the edit distance function as

$$-\log P(Q|C) \propto \text{EditDist}(Q, C) \quad (3)$$

The decoder uses a standard two-pass algorithm to generate 20-best candidates. The first pass uses the Viterbi algorithm to find the best C according to the model of Equations (2) and (3). In the second pass, the A-Star algorithm is used to find the 20-best corrections, using the Viterbi scores computed at each state in the first pass as heuristics. Notice that we always include the input query Q in the 20-best candidate list.

The core of the second component of the speller system is a ranker, which re-ranks the 20-best candidate spelling corrections. If the top C after re-ranking is different than the original query Q, the system returns C as the correction.

Let $\mathbf{f}$ be a feature vector extracted from a query and candidate spelling correction pair (Q, C). The ranker maps $\mathbf{f}$ to a real value y that indicates how likely C is a desired correction of Q. For example, a linear ranker simply maps $\mathbf{f}$ to y with a learned weight vector $\mathbf{w}$ such as $y = \mathbf{w} \cdot \mathbf{f}$, where $\mathbf{w}$ is optimized w.r.t. accuracy on a set of hu-

C:	"disney theme park"	*correct query*
S:	["disney", "theme park"]	*segmentation*
T:	["disnee", "theme part"]	*translation*
M:	(1 → 2, 2→ 1)	*permutation*
Q:	"theme part disnee"	*misspelled query*

Figure 2: Example demonstrating the generative procedure behind the phrase-based error model.

man-labeled (Q, C) pairs. The features in $\mathbf{f}$ are arbitrary functions that map (Q, C) to a real value. Since we define the logarithm of the probabilities of the language model and the error model (i.e., the edit distance function) as features, the ranker can be viewed as a more general framework, subsuming the source channel model as a special case. In our experiments we used 96 features and a non-linear model, implemented as a two-layer neural net, though the details of the ranker and the features are beyond the scope of this paper.

5 A Phrase-Based Error Model

The goal of the phrase-based error model is to transform a correctly spelled query C into a misspelled query Q. Rather than replacing single words in isolation, this model replaces sequences of words with sequences of words, thus incorporating contextual information. For instance, we might learn that "*theme part*" can be replaced by "*theme park*" with relatively high probability, even though "*part*" is not a misspelled word. We assume the following generative story: first the correctly spelled query C is broken into K non-empty word sequences $\mathbf{c}_1, \ldots, \mathbf{c}_k$, then each is replaced with a new non-empty word sequence $\mathbf{q}_1, \ldots, \mathbf{q}_k$, and finally these phrases are permuted and concatenated to form the misspelled Q. Here, $\mathbf{c}$ and $\mathbf{q}$ denote consecutive sequences of words.

To formalize this generative process, let S denote the segmentation of C into K phrases $\mathbf{c}_{1\ldots}\mathbf{c}_K$, and let T denote the K replacement phrases $\mathbf{q}_1\ldots\mathbf{q}_K$ – we refer to these ($\mathbf{c}_i$, $\mathbf{q}_i$) pairs as *bi-phrases*. Finally, let M denote a permutation of K elements representing the final reordering step. Figure 2 demonstrates the generative procedure.

Next let us place a probability distribution over rewrite pairs. Let $B(C, Q)$ denote the set of S, T, M triples that transform C into Q. If we assume a uniform probability over segmentations, then the phrase-based probability can be defined as:

$$P(Q|C) \propto \sum_{\substack{(S,T,M)\in \\ B(C,Q)}} P(T|C,S) \cdot P(M|C,S,T) \quad (4)$$

As is common practice in SMT, we use the maximum approximation to the sum:

$$P(Q|C) \approx \max_{\substack{(S,T,M)\in \\ B(C,Q)}} P(T|C,S) \cdot P(M|C,S,T) \quad (5)$$

5.1 Forced Alignments

Although we have defined a generative model for transforming queries, our goal is not to propose new queries, but rather to provide scores over existing Q and C pairs which act as features for the ranker. Furthermore, the word-level alignments between Q and C can most often be identified with little ambiguity. Thus we restrict our attention to those phrase transformations consistent with a good word-level alignment.

Let J be the length of Q, L be the length of C, and $A = a_1, \ldots, a_J$ be a hidden variable representing the word alignment. Each a_i takes on a value ranging from 1 to L indicating its corresponding word position in C, or 0 if the ith word in Q is unaligned. The cost of assigning k to a_i is equal to the Levenshtein edit distance (Levenshtein, 1966) between the i^{th} word in Q and the k^{th} word in C, and the cost of assigning 0 to a_i is equal to the length of the i^{th} word in Q. We can determine the least cost alignment A^* between Q and C efficiently using the A-star algorithm.

When scoring a given candidate pair, we further restrict our attention to those S, T, M triples that are consistent with the word alignment, which we denote as $B(C, Q, A^*)$. Here, consistency requires that if two words are aligned in A^*, then they must appear in the same bi-phrase ($\mathbf{c}_i$, $\mathbf{q}_i$). Once the word alignment is fixed, the final permutation is uniquely determined, so we can safely discard that factor. Thus we have:

$$P(Q|C) \approx \max_{\substack{(S,T,M)\in \\ \mathcal{B}(C,Q,A^*)}} P(T|C,S) \quad (6)$$

For the sole remaining factor $P(T|C, S)$, we make the assumption that a segmented query $T = \mathbf{q}_1\ldots\ \mathbf{q}_K$ is generated from left to right by transforming each phrase $\mathbf{c}_1\ldots\mathbf{c}_K$ independently:

```
Input: biPhraseLattice "PL" with length = K & height = L;
Initialization: biPhrase.maxProb = 0;
for (x = 0; x <= K – 1; x++)
    for (y = 1; y <= L; y++)
        for (yPre = 1; yPre <= L; yPre++)
        {
            xPre = x – y;
            biPhrasePre = PL.get(xPre, yPre);
            biPhrase = PL.get(x, y);
            if (!biPhrasePre || !biPhrase)
                continue;
            probIncrs = PL.getProbIncrease(biPhrasePre,
                                           biPhrase);
            maxProbPre = biPhrasePre.maxProb;
            totalProb = probIncrs + maxProbPre;
            if (totalProb > biPhrase.maxProb)
            {
                biPhrase.maxProb = totalProb;
                biPhrase.yPre = yPre;
            }
        }
Result: record at each bi-phrase boundary its maximum probability (biPhrase.maxProb) and optimal back-tracking biPhrases (biPhrase.yPre).
```

Figure 3: The dynamic programming algorithm for Viterbi bi-phrase segmentation.

$$P(T|C,S) = \prod_{k=1}^{K} P(\mathbf{q}_k|\mathbf{c}_k), \quad (7)$$

where $P(\mathbf{q}_k|\mathbf{c}_k)$ is a phrase transformation probability, the estimation of which will be described in Section 5.2.

To find the maximum probability assignment efficiently, we can use a dynamic programming approach, somewhat similar to the monotone decoding algorithm described in Och (2002). Here, though, both the input and the output word sequences are specified as the input to the algorithm, as is the word alignment. We define the quantity α_j to be the probability of the most likely sequence of bi-phrases that produce the first j terms of Q and are consistent with the word alignment and C. It can be calculated using the following algorithm:

1. Initialization:

$$\alpha_0 = 1 \quad (8)$$

2. Induction:

$$\alpha_j = \max_{j' < j, \mathbf{q} = q_{j'+1} \ldots q_j} \left\{ \alpha_{j'} P(\mathbf{q}|\mathbf{c}_\mathbf{q}) \right\} \quad (9)$$

3. Total:

$$P(Q|C) = \alpha_J \quad (10)$$

	A	B	C	D	E	F
a	#					
d				#		
c			#			
f						#

a	A
adc	ABCD
d	D
dc	CD
dcf	CDEF
c	C
f	F

Figure 4: Toy example of (left) a word alignment between two strings "adcf" and "ABCDEF"; and (right) the bi-phrases containing up to four words that are consistent with the word alignment.

The pseudo-code of the above algorithm is shown in Figure 3. After generating Q from left to right according to Equations (8) to (10), we record at each possible bi-phrase boundary its maximum probability, and we obtain the total probability at the end-position of Q. Then, by back-tracking the most probable bi-phrase boundaries, we obtain B^*. The algorithm takes a complexity of $O(KL^2)$, where K is the total number of word alignments in A^* which does not contain empty words, and L is the maximum length of a bi-phrase, which is a hyper-parameter of the algorithm. Notice that when we set L=1, the phrase-based error model is reduced to a word-based error model which assumes that words are transformed independently from C to Q, without taking into account any contextual information.

5.2 Model Estimation

We follow a method commonly used in SMT (Koehn et al., 2003) to extract bi-phrases and estimate their replacement probabilities. From each query-correction pair with its word alignment (Q, C, A^*), all bi-phrases consistent with the word alignment are identified. Consistency here implies two things. First, there must be at least one aligned word pair in the bi-phrase. Second, there must not be any word alignments from words inside the bi-phrase to words outside the bi-phrase. That is, we do not extract a phrase pair if there is an alignment from within the phrase pair to outside the phrase pair. The toy example shown in Figure 4 illustrates the bilingual phrases we can generate by this process.

After gathering all such bi-phrases from the full training data, we can estimate conditional relative frequency estimates without smoothing. For example, the phrase transformation probability $P(\mathbf{q}|\mathbf{c})$ in Equation (7) can be estimated approximately as

$$P(\mathbf{q}|\mathbf{c}) = \frac{N(\mathbf{c},\mathbf{q})}{\sum_{\mathbf{q}'} N(\mathbf{c},\mathbf{q}')} \quad (11)$$

where $N(\mathbf{c},\mathbf{q})$ is the number of times that **c** is aligned to **q** in training data. These estimates are useful for contextual lexical selection with sufficient training data, but can be subject to data sparsity issues.

An alternate translation probability estimate not subject to data sparsity issues is the so-called *lexical weight* estimate (Koehn et al., 2003). Assume we have a word translation distribution $t(q|c)$ (defined over individual words, not phrases), and a word alignment A between **q** and **c**; here, the word alignment contains (i, j) pairs, where $i \in 1..|\boldsymbol{q}|$ and $j \in 0..|\mathbf{c}|$, with 0 indicating an inserted word. Then we can use the following estimate:

$$P_w(\mathbf{q}|\mathbf{c},A) = \prod_{i=1}^{|\mathbf{q}|} \frac{1}{|\{j|(j,i)\in A\}|} \sum_{\forall(i,j)\in A} t(q_i|c_j) \quad (12)$$

We assume that for every position in **q**, there is either a single alignment to 0, or multiple alignments to non-zero positions in **c**. In effect, this computes a product of per-word translation scores; the per-word scores are averages of all the translations for the alignment links of that word. We estimate the word translation probabilities using counts from the word aligned corpus: $t(q|c) = \frac{N(c,q)}{\sum_{q'} N(c,q')}$. Here $N(c,q)$ is the number of times that the words (not phrases as in Equation 11) c and q are aligned in the training data. These word based scores of bi-phrases, though not as effective in contextual selection, are more robust to noise and sparsity.

Throughout this section, we have approached this model in a noisy channel approach, finding probabilities of the misspelled query given the corrected query. However, the method can be run in both directions, and in practice SMT systems benefit from also including the direct probability of the corrected query given this misspelled query (Och, 2002).

5.3 Phrase-Based Error Model Features

To use the phrase-based error model for spelling correction, we derive five features and integrate them into the ranker-based query speller system, described in Section 4. These features are as follows.

- **Two phrase transformation features:** These are the phrase transformation scores based on relative frequency estimates in two directions. In the correction-to-query direction, we define the feature as $f_{PT}(Q,C,A) = \log P(Q|C)$, where $P(Q|C)$ is computed by Equations (8) to (10), and $P(\mathbf{q}|\mathbf{c_q})$ is the relative frequency estimate of Equation (11).
- **Two lexical weight features:** These are the phrase transformation scores based on the lexical weighting models in two directions. For example, in the correction-to-query direction, we define the feature as $f_{LW}(Q,C,A) = \log P(Q|C)$, where $P(Q|C)$ is computed by Equations (8) to (10), and the phrase transformation probability is the computed as lexical weight according to Equation (12).
- **Unaligned word penalty feature:** the feature is defined as the ratio between the number of unaligned query words and the total number of query words.

6 Experiments

We evaluate the spelling error models on a large scale real world data set containing 24,172 queries sampled from one year's worth of query logs from a commercial search engine. The spelling of each query is judged and corrected by four annotators.

We divided the data set into training and test data sets. The two data sets do not overlap. The training data contains 8,515 query-correction pairs, among which 1,743 queries are misspelled (i.e., in these pairs, the corrections are different from the queries). The test data contains 15,657 query-correction pairs, among which 2,960 queries are misspelled. The average length of queries in the training and test data is 2.7 words.

The speller systems we developed in this study are evaluated using the following three metrics.

- **Accuracy:** The number of correct outputs generated by the system divided by the total number of queries in the test set.
- **Precision:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of corrections generated by the system.
- **Recall:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of misspelled queries in the test set.

We also perform a significance test, i.e., a t-test with a significance level of 0.05. A significant difference should be read as significant at the 95% level.

#	System	Accuracy	Precision	Recall
1	Source-channel	0.8526	0.7213	0.3586
2	Ranker-based	0.8904	0.7414	0.4964
3	Word model	0.8994	0.7709	0.5413
4	Phrase model (L=3)	0.9043	0.7814	0.5732

Table 1. Summary of spelling correction results.

#	System	Accuracy	Precision	Recall
5	Phrase model (L=1)	0.8994	0.7709	0.5413
6	Phrase model (L=2)	0.9014	0.7795	0.5605
7	Phrase model (L=3)	0.9043	0.7814	0.5732
8	Phrase model (L=5)	0.9035	0.7834	0.5698
9	Phrase model (L=8)	0.9033	0.7821	0.5713

Table 2. Variations of spelling performance as a function of phrase length.

#	System	Accuracy	Precision	Recall
10	L=3; 0 month data	0.8904	0.7414	0.4964
11	L=3; 0.5 month data	0.8959	0.7701	0.5234
12	L=3; 1.5 month data	0.9023	0.7787	0.5667
13	L=3; 3 month data	0.9043	0.7814	0.5732

Table 3. Variations of spelling performance as a function of the size of clickthrough data used for training.

In our experiments, all the speller systems are ranker-based. In most cases, other than the baseline system (a linear neural net), the ranker is a two-layer neural net with 5 hidden nodes. The free parameters of the neural net are trained to optimize accuracy on the training data using the back propagation algorithm, running for 500 iterations with a very small learning rate (0.1) to avoid overfitting. We did not adjust the neural net structure (e.g., the number of hidden nodes) or any training parameters for different speller systems. Neither did we try to seek the best tradeoff between precision and recall. Since all the systems are optimized for accuracy, we use accuracy as the primary metric for comparison.

Table 1 summarizes the main spelling correction results. Row 1 is the baseline speller system where the source-channel model of Equations (2) and (3) is used. In our implementation, we use a linear ranker with only two features, derived respectively from the language model and the error model models. The error model is based on the edit distance function. Row 2 is the ranker-based spelling system that uses all 96 ranking features, as described in Section 4. Note that the system uses the features derived from two error models. One is the edit distance model used for candidate generation. The other is a phonetic model that measures the edit distance between the metaphones (Philips, 1990) of a query word and its aligned correction word. Row 3 is the same system as Row 2, with an additional set of features derived from a word-based error model. This model is a special case of the phrase-based error model described in Section 5 with the maximum phrase length set to one. Row 4 is the system that uses the additional 5 features derived from the phrase-based error models with a maximum bi-phrase length of 3.

In phrase based error model, L is the maximum length of a bi-phrase (Figure 3). This value is important for the spelling performance. We perform experiments to study the impact of L; the results are displayed in Table 2. Moreover, since we proposed to use clickthrough data for spelling correction, it is interesting to study the impact on spelling performance from the size of clickthrough data used for training. We varied the size of clickthrough data and the experimental results are presented in Table 3.

The results show first and foremost that the ranker-based system significantly outperforms the spelling system based solely on the source-channel model, largely due to the richer set of features used (Row 1 vs. Row 2). Second, the error model learned from clickthrough data leads to significant improvements (Rows 3 and 4 vs. Row 2). The phrase-based error model, due to its capability of capturing contextual information, outperforms the word-based model with a small but statistically significant margin (Row 4 vs. Row 3), though using phrases longer ($L > 3$) does not lead to further significant improvement (Rows 6 and 7 vs. Rows 8 and 9). Finally, using more clickthrough data leads to significant improvement (Row 13 vs. Rows 10 to 12). The benefit does not appear to have peaked – further improvements are likely given a larger data set.

7 Conclusions

Unlike conventional textual documents, most search queries consist of a sequence of key words, many of which are valid search terms but are not stored in any compiled lexicon. This presents a challenge to any speller system that is based on a dictionary.

This paper extends the recent research on using Web data and query logs for query spelling correction in two aspects. First, we show that a large amount of training data (i.e. query-correction pairs) can be extracted from clickthrough data, focusing on query reformulation sessions. The resulting data are very clean and effective for error model training. Second, we argue that it is critical to capture contextual information for query spelling correction. To this end, we propose

a new phrase-based error model, which leads to significant improvement in our spelling correction experiments.

There is additional potentially useful information that can be exploited in this type of model. For example, in future work we plan to investigate the combination of the clickthrough data collected from a Web browser with the noisy but large query sessions collected from a commercial search engine.

Acknowledgments

The authors would like to thank Andreas Bode, Mei Li, Chenyu Yan and Galen Andrew for the very helpful discussions and collaboration.

References

Agichtein, E., Brill, E. and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pp. 19-26.

Ahmad, F., and Kondrak, G. 2005. Learning a spelling error model from search query logs. In *HLT-EMNLP*, pp 955-962.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *ACL*, pp. 286-293.

Chen, Q., Li, M., and Zhou, M. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pp. 181-189.

Church, K., Hard, T., and Gao, J. 2007. Compressing trigram language models with Golomb coding. In *EMNLP-CoNLL*, pp. 199-207.

Cucerzan, S., and Brill, E. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP*, pp. 293-300.

Gao, J., Yuan, W., Li, X., Deng, K., and Nie, J-Y. 2009. Smoothing clickthrough data for web search ranking. In *SIGIR*.

Golding, A. R., and Roth, D. 1996. Applying winnow to context-sensitive spelling correction. In *ICML*, pp. 182-190.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*, pp. 133-142.

Kernighan, M. D., Church, K. W., and Gale, W. A. 1990. A spelling correction program based on a noisy channel model. In *COLING*, pp. 205-210.

Koehn, P., Och, F., and Marcu, D. 2003. Statistical phrase-based translation. In *HLT/NAACL*, pp. 127-133.

Kukich, K. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*. 24(4): 377-439.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707-710.

Li, M., Zhu, M., Zhang, Y., and Zhou, M. 2006. Exploring distributional similarity based models for query spelling correction. In *ACL*, pp. 1025-1032.

Mangu, L., and Brill, E. 1997. Automatic rule acquisition for spelling correction. In *ICML*, pp. 187-194.

Och, F. 2002. *Statistical machine translation: from single-word models to alignment templates.* PhD thesis, RWTH Aachen.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4): 417-449.

Okazaki, N., Tsuruoka, Y., Ananiadou, S., and Tsujii, J. 2008. A discriminative candidate generator for string transformations. In *EMNLP,* pp. 447-456.

Philips, L. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):38-44.

Suzuki, H., Li, X., and Gao, J. 2009. Discovery of term variation in Japanese web search queries. In *EMNLP*.

Toutanova, K., and Moore, R. 2002. Pronunciation modeling for improved spelling correction. In *ACL*, pp. 144-151.

Wang, X., and Zhai, C. 2008. Mining term association patterns from search logs for effective query reformulation. In *CIKM*, pp. 479-488.

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis, G. 2009. Using the web for language independent spellchecking and autocorrection. In *EMNLP*, pp. 890-899.

Inducing Domain-specific Semantic Class Taggers from (Almost) Nothing

Ruihong Huang and Ellen Riloff
School of Computing
University of Utah
Salt Lake City, UT 84112
{huangrh,riloff}@cs.utah.edu

Abstract

This research explores the idea of inducing domain-specific semantic class taggers using only a domain-specific text collection and seed words. The learning process begins by inducing a classifier that only has access to contextual features, forcing it to generalize beyond the seeds. The contextual classifier then labels new instances, to expand and diversify the training set. Next, a *cross-category bootstrapping process* simultaneously trains a suite of classifiers for multiple semantic classes. The positive instances for one class are used as negative instances for the others in an iterative bootstrapping cycle. We also explore a one-semantic-class-per-discourse heuristic, and use the classifiers to dynamically create semantic features. We evaluate our approach by inducing six semantic taggers from a collection of veterinary medicine message board posts.

1 Introduction

The goal of our research is to create semantic class taggers that can assign a semantic class label to every noun phrase in a sentence. For example, consider the sentence: *"The lab mix was diagnosed with parvo and given abx"*. A semantic tagger should identify the *"the lab mix"* as an ANIMAL, *"parvo"* as a DISEASE, and *"abx"* (antibiotics) as a DRUG. Accurate semantic tagging could be beneficial for many NLP tasks, including coreference resolution and word sense disambiguation, and many NLP applications, such as event extraction systems and question answering technology.

Semantic class tagging has been the subject of previous research, primarily under the guises of *named entity recognition* (NER) and *mention detection*. Named entity recognizers perform semantic tagging on proper name noun phrases, and sometimes temporal and numeric expressions as well. The *mention detection* task was introduced in recent ACE evaluations (e.g., (ACE, 2007; ACE, 2008)) and requires systems to identify all noun phrases (proper names, nominals, and pronouns) that correspond to 5-7 semantic classes.

Despite widespread interest in semantic tagging, nearly all semantic taggers for comprehensive NP tagging still rely on supervised learning, which requires annotated data for training. A few annotated corpora exist, but they are relatively small and most were developed for broad-coverage NLP. Many domains, however, are replete with specialized terminology and jargon that cannot be adequately handled by general-purpose systems. Domains such as biology, medicine, and law are teeming with specialized vocabulary that necessitates training on domain-specific corpora.

Our research explores the idea of inducing domain-specific semantic taggers using a small set of seed words as the only form of human supervision. Given an (unannotated) collection of domain-specific text, we automatically generate training instances by labelling every instance of a seed word with its designated semantic class. We then train a classifier to do semantic tagging using these seed-based annotations, using bootstrapping to iteratively improve performance.

On the surface, this approach appears to be a contradiction. The classifier must learn how to assign different semantic tags to different instances of the same word based on context (e.g., *"lab"* may refer to an animal in one context but a laboratory in another). And yet, we plan to train the classifier using stand-alone seed words, making the assumption that every instance of the seed belongs to the same semantic class. We resolve this apparent contradiction by using semantically unambiguous seeds and by introducing an initial context-only training phase before bootstrapping begins. First, we train a *strictly contextual* classifier that only

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 275–285,
Uppsala, Sweden, 11-16 July 2010.

has access to contextual features and cannot see the seed. Then we apply the classifier to the corpus to automatically label new instances, and combine these new instances with the seed-based instances. This process expands and diversifies the training set to fuel subsequent bootstrapping.

Another challenge is that we want to use a *small* set of seeds to minimize the amount of human effort, and then use bootstrapping to fully exploit the domain-specific corpus. Iterative self-training, however, often has difficulty sustaining momentum or it succumbs to semantic drift (Komachi et al., 2008; McIntosh and Curran, 2009). To address these issues, we simultaneously induce a suite of classifiers for multiple semantic categories, using the positive instances of one semantic category as negative instances for the others. As bootstrapping progresses, the classifiers gradually improve themselves, and each other, over many iterations. We also explore a *one-semantic-class-per-discourse* (OSCPD) heuristic that infuses the learning process with fresh training instances, which may be substantially different from the ones seen previously, and we use the labels produced by the classifiers to dynamically create semantic features.

We evaluate our approach by creating six semantic taggers using a collection of message board posts in the domain of veterinary medicine. Our results show this approach produces high-quality semantic taggers after a sustained bootstrapping cycle that maintains good precision while steadily increasing recall over many iterations.

2 Related Work

Semantic class tagging is most closely related to *named entity recognition* (NER), *mention detection*, and *semantic lexicon induction*. NER systems (e.g., (Bikel et al., 1997; Collins and Singer, 1999; Cucerzan and Yarowsky, 1999; Fleischman and Hovy, 2002) identify proper named entities, such as people, organizations, and locations. Several bootstrapping methods for NER have been previously developed (e.g., (Collins and Singer, 1999; Niu et al., 2003)). NER systems, however, do not identify nominal NP instances (e.g., "a software manufacturer" or "the beach"), or handle semantic classes that are not associated with proper named entities (e.g., symptoms).[1] ACE mention detection systems (e.g., see (ACE, 2005; ACE, 2007; ACE, 2008)) require tagging of NPs that correspond to 5-7 general semantic classes. These systems are typically trained with supervised learning using annotated corpora, although techniques have been developed to use resources for one language to train systems for different languages (e.g., (Zitouni and Florian, 2009)).

Another line of relevant work is *semantic class induction* (e.g., (Riloff and Shepherd, 1997; Roark and Charniak, 1998; Thelen and Riloff, 2002; Ng, 2007; McIntosh and Curran, 2009), where the goal is to induce a stand-alone dictionary of words with semantic class labels. These techniques are often designed to learn specialized terminology from unannotated domain-specific texts via bootstrapping. Our work, however, focuses on classification of NP *instances* in context, so the same phrase may be assigned to different semantic classes in different contexts. Consequently, our classifier can also assign semantic class labels to pronouns.

There has also been work on extracting semantically related terms or category members from the Web (e.g., (Paşca, 2004; Etzioni et al., 2005; Kozareva et al., 2008; Carlson et al., 2009)). These techniques harvest broad-coverage semantic information from the Web using patterns and statistics, typically for the purpose of knowledge acquisition. Importantly, our goal is to classify instances in context, rather than generate lists of terms. In addition, the goal of our research is to learn specialized terms and jargon that may not be common on the Web, as well as domain-specific usages that may differ from the norm (e.g., *"mix"* and *"lab"* are usually ANIMALS in our domain).

The idea of simulataneously learning multiple semantic categories to prevent semantic drift has been explored for other tasks, such as semantic lexicon induction (Thelen and Riloff, 2002; McIntosh and Curran, 2009) and pattern learning (Yangarber, 2003). Our bootstrapping model can be viewed as a form of self-training (e.g., (Ng and Cardie, 2003; Mihalcea, 2004; McClosky et al., 2006)), and cross-category training is similar in spirit to co-training (e.g., (Blum and Mitchell, 1998; Collins and Singer, 1999; Riloff and Jones, 1999; Mueller et al., 2002; Phillips and Riloff, 2002)). But, importantly, our classifiers all use the same feature set so they do not represent independent views of the data. They do, however, offer slightly different perspectives because each is at-

[1] Some NER systems also handle specialized constructs such as dates and monetary amounts.

tempting to recognize a different semantic class.

3 Bootstrapping an Instance-based Semantic Class Tagger from Seeds

3.1 Motivation

Our goal is to create a bootstrapping model that can rapidly create semantic class taggers using just a small set of seed words and an unannotated domain-specific corpus. Our motivation comes from specialized domains that cannot be adequately handled by general-purpose NLP systems. As an example of such a domain, we have been working with a collection of message board posts in the field of veterinary medicine. Given a document, we want a semantic class tagger to label every NP with a semantic category, for example:

> [A 14yo doxy]$_{ANIMAL}$ owned by [a reputable breeder]$_{HUMAN}$ is being treated for [IBD]$_{DISEASE}$ with [pred]$_{DRUG}$.

When we began working with these texts, we were immediately confronted by a dizzying array of non-standard words and word uses. In addition to formal veterinary vocabulary (e.g., animal diseases), veterinarians often use informal, shorthand terms when posting on-line. For example, they frequently refer to breeds using "nicknames" or shortened terms (e.g., *gshep* for German shepherd, *doxy* for dachsund, *bxr* for boxer, *labx* for labrador cross). Often, they refer to animals based solely on their physical characteristics, for example *"a dlh"* (domestic long hair), *"a m/n"* (male, neutered), or *"a 2yo"* (2 year old). This phenomenon occurs with other semantic categories as well, such as drugs and medical tests (e.g., *pred* for prednisone, and *rads* for radiographs).

Nearly all semantic class taggers are trained using supervised learning with manually annotated data. However, annotated data is rarely available for specialized domains, and it is expensive to obtain because domain experts must do the annotation work. So we set out to create a bootstrapping model that can rapidly create domain-specific semantic taggers using just a few seed words and a domain-specific text collection.

Our bootstrapping model consists of two distinct phases. First, we train *strictly contextual classifiers* from the seed annotations. We then apply the classifiers to the unlabeled data to generate new annotated instances that are added to the training set. Second, we employ a *cross-category bootstrapping* process that simultaneously trains a suite of classifiers for multiple semantic categories, using the positive instances for one semantic class as negative instances for the others. This cross-category training process gives the learner sustained momentum over many bootstrapping iterations. Finally, we explore two additional enhancements: (1) a *one-semantic-class-per-discourse* heuristic to automatically generate new training instances, and (2) dynamically created semantic features produced by the classifiers themselves. In the following sections, we explain each of these steps in detail.

3.2 Phase 1: Inducing a Contextual Classifier

The main challenge that we faced was how to train an instance-based classifier using seed words as the only form of human supervision. First, the user must provide a small set of seed words that are relatively unambiguous (e.g., *"dog"* will nearly always refer to an animal in our domain). But even so, training a traditional classifier from seed-based instances would likely produce a classifier that learns to recognize the seeds but is unable to classify new examples. We needed to force the classifier to generalize beyond the seed words.

Our solution was to introduce an initial training step that induces a *strictly contextual classifier*. First, we generate training instances by automatically labeling each instance of a seed word with its designated semantic class. However, when we create feature vectors for the classifier, the seeds themselves are hidden and only contextual features are used to represent each training instance. By essentially "masking" the seed words so the classifier can only see the contexts around them, we force the classifier to generalize.

We create a suite of *strictly contextual classifiers*, one for each semantic category. Each classifier makes a binary decision as to whether a noun phrase belongs to its semantic category. We use the seed words for category C_k to generate positive training instances for the C_k classifier, and the seed words for all other categories to generate the negative training instances for C_k.

We use an in-house sentence segmenter and NP chunker to identify the base NPs in each sentence and create feature vectors that represent each constituent in the sentence as either an NP or an individual word. For each seed word, the feature

vector captures a context window of 3 constituents (word or NP) to its left and 3 constituents (word or NP) to its right. Each constituent is represented with a lexical feature: for NPs, we use its head noun; for individual words, we use the word itself. The seed word, however, is discarded so that the classifier is essentially blind-folded and cannot see the seed that produced the training instance. We also create a feature for every modifier that precedes the head noun in the target NP, except for articles which are discarded. As an example, consider the following sentence:

> *Fluffy was diagnosed with FELV after a blood test showed that he tested positive.*

Suppose that "*FELV*" is a seed for the DISEASE category and "*test*" is a seed for the TEST category. Two training instances would be created, with feature vectors that look like this, where M represents a modifier inside the target NP:

was_{-3} $diagnosed_{-2}$ $with_{-1}$ $after_1$ $test_2$ $showed_3$ $\Rightarrow$ DISEASE

$with_{-3}$ $FELV_{-2}$ $after_{-1}$ $blood_M$ $showed_1$ $that_2$ $he_3 \Rightarrow$ TEST

The contextual classifiers are then applied to the corpus to automatically label new instances. We use a confidence score to label only the instances that the classifiers are most certain about. We compute a confidence score for instance i with respect to semantic class C_k by considering both the score of the C_k classifier as well as the scores of the competing classifiers. Intuitively, we have confidence in labeling an instance as category C_k if the C_k classifier gave it a positive score, and its score is much higher than the score of any other classifier. We use the following scoring function:

Confidence(i,C_k) =
score(i,C_k) - max($\forall_{j \neq k}$ score(i,C_j))

We employ support vector machines (SVMs) (Joachims, 1999) with a linear kernel as our classifiers, using the SVMlin software (Keerthi and DeCoste, 2005). We use the value produced by the decision function (essentially the distance from the hyperplane) as the score for a classifier. We specify a threshold θ_{cf} and only assign a semantic tag C_k to an instance i if Confidence(i,C_k) $\geq \theta_{cf}$.

All instances that pass the confidence threshold are labeled and added to the training set. This process greatly enhances the diversity of the training data. In this initial learning step, the strictly contextual classifiers substantially increase the number of training instances for each semantic category, producing a more diverse mix of seed-generated instances and context-generated instances.

3.3 Phase 2: Cross-Category Bootstrapping

The next phase of the learning process is an iterative bootstrapping procedure. The key challenge was to design a bootstrapping model that would not succumb to semantic drift and would have sustained momentum to continue learning over many iterations.

Figure 1 shows the design of our *cross-category bootstrapping* model.[2] We simultaneously train a suite of binary classifiers, one for each semantic category, $C_1 \ldots C_n$. After each training cycle, all of the classifiers are applied to the remaining unlabeled instances and each classifier labels the (positive) instances that it is most confident about (i.e., the instances that it classifies with a confidence score $\geq \theta_{cf}$). The set of instances positively labeled by classifier C_k are shown as C_k^+ in Figure 1. All of the new instances produced by classifier C_k are then added to the set of positive training instances for C_k and to the set of negative training instances for all of the other classifiers.

One potential problem with this scheme is that some categories are more prolific than others, plus we are collecting negative instances from a set of competing classifiers. Consequently, this approach can produce highly imbalanced training sets. Therefore we enforced a 3:1 ratio of negatives to positives by randomly selecting a subset of the possible negative instances. We discuss this issue further in Section 4.4.

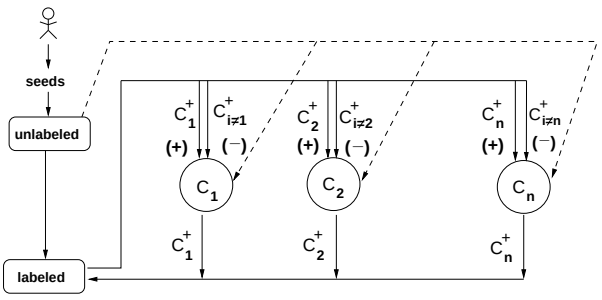


Figure 1: Cross-Category Bootstrapping

[2] For simplicity, this picture does not depict the initial contextual training step, but that can be viewed as the first iteration in this general framework.

Cross-category training has two advantages over independent self-training. First, as others have shown for pattern learning and lexicon induction (Thelen and Riloff, 2002; Yangarber, 2003; McIntosh and Curran, 2009), simultaneously training classifiers for multiple categories reduces semantic drift because each classifier is deterred from encroaching on another one's territory (i.e., claiming the instances from a competing class as its own). Second, similar in spirit to co-training[3], this approach allows each classifier to obtain new training instances from an outside source that has a slightly different perspective. While independent self-training can quickly run out of steam, cross-category training supplies each classifier with a constant stream of new (negative) instances produced by competing classifiers. In Section 4, we will show that cross-category bootstrapping performs substantially better than an independent self-training model, where each classifier is bootstrapped separately.

The feature set for these classifiers is exactly the same as described in Section 3.2, except that we add a new lexical feature that represents the head noun of the target NP (i.e., the NP that needs to be tagged). This allows the classifiers to consider the local context as well as the target word itself when making decisions.

3.4 One Semantic Class Per Discourse

We also explored the idea of using a *one semantic class per discourse (OSCPD)* heuristic to generate additional training instances during bootstrapping. Inspired by Yarowsky's *one sense per discourse* heuristic for word sense disambiguation (Yarowsky, 1995), we make the assumption that multiple instances of a word in the same discourse will nearly always correspond to the same semantic class. Since our data set consists of message board posts organized as threads, we consider all posts in the same thread to be a single discourse.

After each training step, we apply the classifiers to the unlabeled data to label some new instances. For each newly labeled instance, the OSCPD heuristic collects all instances with the same head noun in the same discourse (thread) and unilaterally labels them with the same semantic class. This heuristic serves as meta-knowledge to label instances that (potentially) occur in very different contexts, thereby infusing the bootstrapping process with "fresh" training examples.

In early experiments, we found that OSCPD can be aggressive, pulling in many new instances. If the classifier labels a word incorrectly, however, then the OSCPD heuristic will compound the error and mislabel even more instances incorrectly. Therefore we only apply this heuristic to instances that are labeled with extremely high confidence ($\theta_{cf} \geq 2.5$) and that pass a global sanity check, $gsc(w) \geq 0.2$, which ensures that a relatively high proportion of labeled instances with the same head noun have been assigned to the same semantic class. Specifically, $gsc(w) = 0.1 * \frac{w_{l/c}}{w_l} + 0.9 * \frac{w_{u/c}}{w_u}$ where w_l and w_u are the # of labeled and unlabeled instances, respectively, $w_{l/c}$ is the # of instances labeled as c, and $w_{u/c}$ is the # of unlabeled instances that receive a positive confidence score for c when given to the classifier. The intuition behind the second term is that most instances are initially unlabeled and we want to make sure that many of the unlabeled instances are likely to belong to the same semantic class (even though the classifier isn't ready to commit to them yet).

3.5 Dynamic Semantic Features

For many NLP tasks, classifiers use semantic features to represent the semantic class of words. These features are typically obtained from external resources such as Wordnet (Miller, 1990). Our bootstrapping model incrementally trains semantic class taggers, so we explored the idea of using the labels assigned by the classifiers to create enhanced feature vectors by dynamically adding semantic features. This process allows later stages of bootstrapping to directly benefit from earlier stages. For example, consider the sentence:

He started the doxy on Vetsulin today.

If *"Vetsulin"* was labeled as a DRUG in a previous bootstrapping iteration, then the feature vector representing the context around *"doxy"* can be enhanced to include an additional semantic feature identifying Vetsulin as a DRUG, which would look like this:

$$He_{-2}\ started_{-1}\ on_{1}\ Vetsulin_{2}\ DRUG_{2}\ today_{3}$$

Intuitively, the semantic features should help the classifier identify more general contextual patterns, such as *"started <X> on DRUG"*. To create semantic features, we use the semantic tags that

[3]But technically this is not co-training because our feature sets are all the same.

have been assigned to the current set of labeled instances. When a feature vector is created for a target NP, we check every noun instance in its context window to see if it has been assigned a semantic tag, and if so, then we add a semantic feature. In the early stages of bootstrapping, however, relatively few nouns will be assigned semantic tags, so these features are often missing.

3.6 Thresholds and Stopping Criterion

When new instances are automatically labeled during bootstrapping, it is critically important that most of the labels are correct or performance rapidly deteriorates. This suggests that we should only label instances in which the classifier has high confidence. On the other hand, a high threshold often yields few new instances, which can cause the bootstrapping process to sputter and halt. To balance these competing demands, we used a *sliding threshold* that begins conservatively but gradually loosens the reins as bootstrapping progresses. Initially, we set $\theta_{cf} = 2.0$, which only labels instances that the classifier is highly confident about. When fewer than min new instances can be labeled, we automatically decrease θ_{cf} by 0.2, allowing another batch of new instances to be labeled, albeit with slightly less confidence. We continue decreasing the threshold, as needed, until $\theta_{cf} < 1.0$, when we end the bootstrapping process. In Section 4, we show that this sliding threshold outperforms fixed threshold values.

4 Evaluation

4.1 Data

Our data set consists of message board posts from the Veterinary Information Network (VIN), which is a web site (www.vin.com) for professionals in veterinary medicine. Among other things, VIN hosts forums where veterinarians engage in discussions about medical issues, cases in their practices, etc. Over half of the small animal veterinarians in the U.S. and Canada use VIN. Analysis of veterinary data could not only improve pet health care, but also provide early warning signs of infectious disease outbreaks, emerging zoonotic diseases, exposures to environmental toxins, and contamination in the food chain.

We obtained over 15,000 VIN message board threads representing three topics: cardiology, endocrinology, and feline internal medicine. We did basic cleaning, removing html tags and tokenizing numbers. For training, we used 4,629 threads, consisting of 25,944 individual posts. We developed classifiers to identify six semantic categories: ANIMAL, DISEASE/SYMPTOM[4], DRUG, HUMAN, TEST, and OTHER.

The message board posts contain an abundance of veterinary terminology and jargon, so two domain experts[5] from VIN created a test set (answer key) for our evaluation. We defined annotation guidelines[6] for each semantic category and conducted an inter-annotator agreement study to measure the consistency of the two domain experts on 30 message board posts, which contained 1,473 noun phrases. The annotators achieved a relatively high κ score of .80. Each annotator then labeled an additional 35 documents, which gave us a *test set* containing 100 manually annotated message board posts. The table below shows the distribution of semantic classes in the test set.

Animal	Dis/Sym	Drug	Test	Human	Other
612	900	369	404	818	1723

To select seed words, we used the procedure proposed by Roark and Charniak (1998), ranking all of the head nouns in the training corpus by frequency and manually selecting the first 10 nouns that *unambiguously* belong to each category.[7] This process is fast, relatively objective, and guaranteed to yield high-frequency terms, which is important for bootstrapping. We used the Stanford part-of-speech tagger (Toutanova et al., 2003) to identify nouns, and our own simple rule-based NP chunker.

4.2 Baselines

To assess the difficulty of our data set and task, we evaluated several baselines. The first baseline searches for each head noun in WordNet and labels the noun as category C_k if it has a hypernym synset corresponding to that category. We manually identified the WordNet synsets that, to the best of our ability, seem to most closely correspond

[4]We used a single category for diseases and symptoms because our domain experts had difficulty distinguishing between them. A veterinary consultant explained that the same term (e.g., diabetes) may be considered a symptom in one context if it is secondary to another condition (e.g., pancreatitis) and a disease in a different context if it is the primary diagnosis.

[5]One annotator is a veterinarian and the other is a veterinary technician.

[6]The annotators were also allowed to label an NP as *POS_Error* if it was clearly misparsed. These cases were not used in the evaluation.

[7]We used 20 seeds for DIS/SYM because we merged two categories and for OTHER because it is a broad catch-all class.

Method	Animal	Dis/Sym	Drug	Test	Human	Other	Avg
BASELINES							
WordNet	32/80/46	21/81/34	25/35/29	NA	62/66/64	NA	35/66/45.8
Seeds	38/100/55	14/99/25	21/97/35	29/94/45	80/99/88	18/93/30	37/98/53.1
Supervised	67/94/78	20/88/33	24/96/39	34/90/49	79/99/88	31/91/46	45/94/60.7
Ind. Self-Train I.13	61/84/71	39/80/52	53/77/62	55/70/61	81/96/88	30/82/44	58/81/67.4
CROSS-CATEGORY BOOTSTRAPPED CLASSIFIERS							
Contextual I.1	59/77/67	33/84/47	42/80/55	49/77/59	82/93/87	33/80/47	53/82/64.3
XCategory I.45	86/71/78	57/82/67	70/78/74	73/65/69	85/92/89	46/82/59	75/78/76.1
XCat+OSCPD I.40	86/69/77	59/81/68	72/70/71	72/69/71	86/92/89	50/81/62	75/76/75.6
XCat+OSCPD+SF I.39	86/70/77	60/81/69	69/81/75	73/69/71	86/91/89	50/81/62	75/78/76.6

Table 1: Experimental results, reported as Recall/Precision/F score

to each semantic class. We do not report WordNet results for TEST because there did not seem be an appropriate synset, or for the OTHER category because that is a catch-all class. The first row of Table 1 shows the results, which are reported as Recall/Precision/F score[8]. The WordNet baseline yields low recall (21-32%) for every category except HUMAN, which confirms that many veterinary terms are not present in WordNet. The surprisingly low precision for some categories is due to atypical word uses (e.g., *patient*, *boy*, and *girl* are HUMAN in WordNet but nearly always ANIMALS in our domain), and overgeneralities (e.g., WordNet lists *calcium* as a DRUG).

The second baseline simply labels every instance of a seed with its designated semantic class. All non-seed instances remain unlabeled. As expected, the seeds produce high precision but low recall. The exception is HUMAN, where 80% of the instances match a seed word, undoubtedly because five of the ten HUMAN seeds are 1st and 2nd person pronouns, which are extremely common.

A third baseline trains semantic classifiers using supervised learning by performing 10-fold cross-validation on the test set. The feature set and classifier settings are exactly the same as with our bootstrapped classifiers.[9] Supervised learning achieves good precision but low recall for all categories except ANIMAL and HUMAN. In the next section, we present the experimental results for our bootstrapped classifiers.

4.3 Results for Bootstrapped Classifiers

The bottom section of Table 1 displays the results for our bootstrapped classifiers. The **Contextual I.1** row shows results after just the first iteration, which trains only the *strictly contextual classifiers*. The average F score improved from 53.1 for the seeds alone to 64.3 with the contextual classifiers. The next row, **XCategory I.45**, shows the results after cross-category bootstrapping, which ended after 45 iterations. (We indicate the number of iterations until bootstrapping ended using the notation **I.#**.) With cross-category bootstrapping, the average F score increased from 64.3 to 76.1. A closer inspection reveals that all of the semantic categories except HUMAN achieved large recall gains. And importantly, these recall gains were obtained with relatively little loss of precision, with the exception of TEST.

Next, we measured the impact of the *one-semantic-class-per-discourse* heuristic, shown as **XCat+OSCPD I.40**. From Table 1, it appears that OSCPD produced mixed results: recall increased by 1-4 points for DIS/SYM, DRUG, HUMAN, and OTHER, but precision was inconsistent, improving by +4 for TEST but dropping by -8 for DRUG. However, this single snapshot in time does not tell the full story. Figure 2 shows the performance of the classifiers during the course of bootstrapping. The OSCPD heuristic produced a steeper learning curve, and consistently improved performance until the last few iterations when its performance dipped. This is probably due to the fact that noise gradually increases during bootstrapping, so incorrect labels are more likely and OSCPD will compound any mistakes by the classifier. A good future strategy might be to use the OSCPD heuristic only during the early stages of bootstrapping when the classifier's decisions are most reliable.

We also evaluated the effect of dynamically created semantic features. When added to the basic XCategory system, they had almost no effect. We suspect this is because the semantic features are sparse during most of the bootstrapping process. However, the semantic features did im-

[8]We use an F(1) score, where recall and precision are equally weighted.

[9]For all of our classifiers, supervised and bootstrapped, we label all instances of the seed words first and then apply the classifiers to the unlabeled (non-seed) instances.

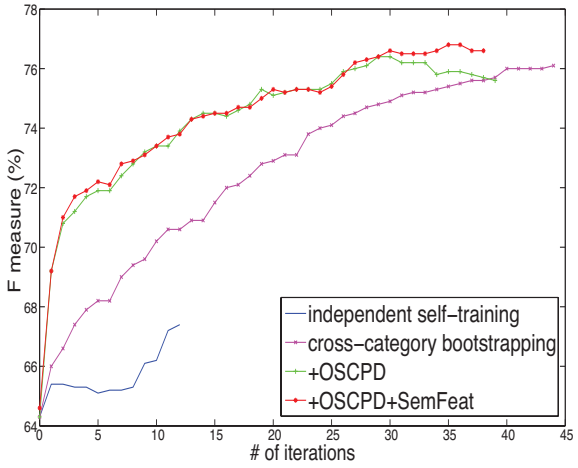


Figure 2: Average F scores after each iteration

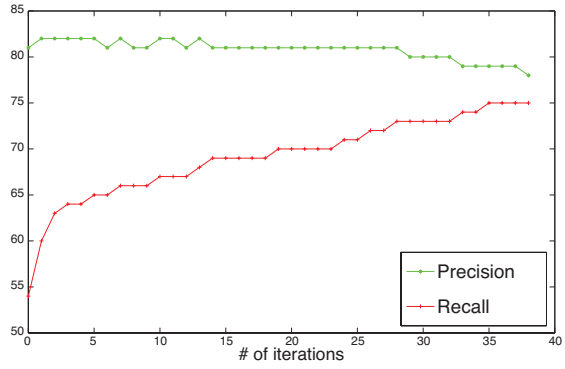


Figure 3: Recall and Precision scores during cross-category bootstrapping

prove performance when coupled with the OSCPD heuristic, presumably because the OSCPD heuristic aggressively labels more instances in the earlier stages of bootstrapping, increasing the prevalence of semantic class tags. The **XCat+OSCPD+SF I.39** row in Table 1 shows that the semantic features coupled with OSCPD dramatically increased the precision for DRUG, yielding the best overall F score of 76.6.

We conducted one additional experiment to assess the benefits of cross-category bootstrapping. We created an analogous suite of classifiers using self-training, where each classifier independently labels the instances that it is most confident about, adds them only to its own training set, and then retrains itself. The **Ind. Self-Train I.13** row in Table 1 shows that these classifiers achieved only 58% recall (compared to 75% for **XCategory**) and an average F score of 67.4 (compared to 76.1 for **XCategory**). One reason for the disparity is that the self-training model ended after just 13 bootstrapping cycles (**I.13**), given the same threshold values. To see if we could push it further, we lowered the confidence threshold to 0 and it continued learning through 35 iterations. Even so, its final score was 65% recall with 79% precision, which is still well below the 75% recall with 78% precision produced by the **XCategory** model. These results support our claim that cross-category bootstrapping is more effective than independently self-trained models.

Figure 3 tracks the recall and precision scores of the **XCat+OSCPD+SF** system as bootstrapping progresses. This graph shows the sustained momentum of cross-category bootstrapping: recall steadily improves while precision stays consistently high with only a slight dropoff at the end.

4.4 Analysis

To assess the impact of corpus size, we generated a learning curve with randomly selected subsets of the training texts. Figure 4 shows the average F score of our best system using $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and all of the data. With just $\frac{1}{16}$th of the training set, the system has about 1,600 message board posts to use for training, which yields a similar F score (roughly 61%) as the supervised baseline that used 100 manually annotated posts via 10-fold cross-validation. So with 16 times more text, seed-based bootstrapping achieves roughly the same results as supervised learning. This result reflects the natural trade-off between supervised learning and seed-based bootstrapping. Supervised learning exploits manually annotated data, but must make do with a relatively small amount of training text because manual annotations are expensive. In contrast, seed-based bootstrapping exploits a small number of human-provided seeds, but needs a larger set of (unannotated) texts for training because the seeds produce relatively sparse annotations of the texts.

An additional advantage of seed-based bootstrapping methods is that they can easily exploit unlimited amounts of training text. For many domains, large text collections are readily available. Figure 4 shows a steady improvement in performance as the amount of training text grows. Overall, the F score improves from roughly 61% to nearly 77% simply by giving the system access to more unannotated text during bootstrapping.

We also evaluated the effectiveness of our sliding confidence threshold (Section 3.6). The table below shows the results using fixed thresholds

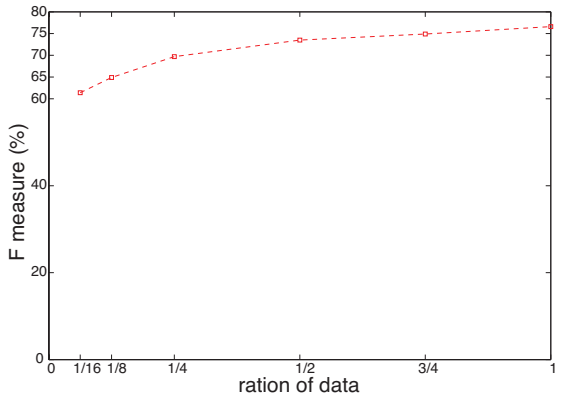


Figure 4: Learning Curve

of 1.0, 1.5, 2.0, as well as the sliding threshold (which begins at 2.0 and ends at 1.0 decreasing by 0.2 when the number of newly labeled instances falls below 3000 (i.e., < 500 per category, on average). This table depicts the expected trade-off between recall and precision for the fixed thresholds, with higher thresholds producing higher precision but lower recall. The sliding threshold produces the best F score, achieving the best balance of high recall and precision.

θ_{cf}	R/P/F
1.0	71/77/74.1
1.5	69/81/74.7
2.0	65/82/72.4
Sliding	75/78/76.6

As mentioned in Section 3.3, we used 3 times as many negative instances as positive instances for every semantic category during bootstrapping. This ratio was based on early experiments where we needed to limit the number of negative instances per category because the cross-category framework naturally produces an extremely skewed negative/positive training set. We revisited this issue to empirically assess the impact of the negative/positive ratio on performance. The table below shows recall, precision, and F score results when we vary the ratio from 1:1 to 5:1. A 1:1 ratio seems to be too conservative, improving precision a bit but lowering recall. However the difference in performance between the other ratios is small. Our conclusion is that a 1:1 ratio is too restrictive but, in general, the cross-category bootstrapping process is relatively insensitive to the specific negative/positive ratio used. Our observation from preliminary experiments, however, is that the negative/positive ratio does need to be controlled, or else the dominant categories overwhelm the less frequent categories with negative instances.

Neg:Pos	R/P/F
1:1	72/79/75.2
2:1	74/78/76.1
3:1	75/78/76.6
4:1	75/77/76.0
5:1	76/77/76.4

Finally, we examined performance on gendered pronouns (he/she/him/her), which can refer to either animals or people in the veterinary domain. 84% (220/261) of the gendered pronouns were annotated as ANIMAL in the test set. Our classifier achieved 95% recall (209/220) and 90% precision (209/232) for ANIMAL and 15% recall (6/41) and 100% precision (6/6) for HUMAN. So while it failed to recognize most of the (relatively few) gendered pronouns that refer to a person, it was highly effective at identifying the ANIMAL references and it was always correct when it did assign a HUMAN tag to a pronoun.

5 Conclusions

We presented a novel technique for inducing domain-specific semantic class taggers from a handful of seed words and an unannotated text collection. Our results showed that the induced taggers achieve good performance on six semantic categories associated with the domain of veterinary medicine. Our technique allows semantic class taggers to be rapidly created for specialized domains with minimal human effort. In future work, we plan to investigate whether these semantic taggers can be used to improve other tasks.

Acknowledgments

We are very grateful to the people at the Veterinary Information Network for providing us access to their resources. Special thanks to Paul Pion, DVM and Nicky Mastin, DVM for making their data available to us, and to Sherri Lofing and Becky Lundgren, DVM for their time and expertise in creating the gold standard annotations. This research was supported in part by Department of Homeland Security Grant N0014-07-1-0152 and Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program.

References

ACE. 2005. NIST ACE evaluation website. In *http://www.nist.gov/speech/tests/ace/2005*.

ACE. 2007. NIST ACE evaluation website. In *http://www.nist.gov/speech/tests/ace/2007*.

ACE. 2008. NIST ACE evaluation website. In *http://www.nist.gov/speech/tests/ace/2008*.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201.

A. Blum and T. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*.

Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *HLT-NAACL 2009 Workshop on Semi-Supervised Learning for NLP*.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*.

S. Cucerzan and D. Yarowsky. 1999. Language Independent Named Entity Recognition Combining Morphologi cal and Contextual Evidence. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

M.B. Fleischman and E.H. Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the COLING conference*, August.

T. Joachims. 1999. Making Large-Scale Support Vector Machine Learning Practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

S. Keerthi and D. DeCoste. 2005. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

Z. Kozareva, E. Riloff, and E. Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08)*.

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

T. McIntosh and J. Curran. 2009. Reducing Semantic Drift with Bagging and Distributional Similarity. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*.

R. Mihalcea. 2004. Co-training and Self-training for Word Sense Disambiguation. In *CoNLL-2004*.

G. Miller. 1990. Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4).

C. Mueller, S. Rapp, and M. Strube. 2002. Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

V. Ng and C. Cardie. 2003. Weakly supervised natural language learning without redundant views. In *HLT-NAACL-2003*.

V. Ng. 2007. Semantic Class Induction and Coreference Resolution. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Cheng Niu, Wei Li, Jihong Ding, and Rohini K. Srihari. 2003. A bootstrapping approach to named entity classification using successive learners. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL-03)*, pages 335–342.

M. Paşca. 2004. Acquisition of categorized named entities for web search. In *Proc. of the Thirteenth ACM International Conference on Information and Knowledge Management*, pages 137–145.

W. Phillips and E. Riloff. 2002. Exploiting Strong Syntactic Heuristics and Co-Training to Learn Semantic Lexicons. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 125–132.

E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

E. Riloff and J. Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

B. Roark and E. Charniak. 1998. Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.

M. Thelen and E. Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pa ttern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003*.

R. Yangarber. 2003. Counter-training in the discovery of semantic patterns. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

Imed Zitouni and Radu Florian. 2009. Cross-language information propagation for arabic mention detection. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):1–21.

Learning 5000 Relational Extractors

Raphael Hoffmann, Congle Zhang, Daniel S. Weld
Computer Science & Engineering
University of Washington
Seattle, WA-98195, USA
{raphaelh,clzhang,weld}@cs.washington.edu

Abstract

Many researchers are trying to use information extraction (IE) to create large-scale knowledge bases from natural language text on the Web. However, the primary approach (supervised learning of relation-specific extractors) requires manually-labeled training data for each relation and doesn't scale to the thousands of relations encoded in Web text.

This paper presents LUCHS, a self-supervised, relation-specific IE system which learns 5025 relations — more than an order of magnitude greater than any previous approach — with an average F1 score of 61%. Crucial to LUCHS's performance is an automated system for *dynamic lexicon learning*, which allows it to learn accurately from heuristically-generated training data, which is often noisy and sparse.

1 Introduction

Information extraction (IE), the process of generating relational data from natural-language text, has gained popularity for its potential applications in Web search, question answering and other tasks. Two main approaches have been attempted:

- Supervised learning of relation-specific extractors (*e.g.*, (Freitag, 1998)), and
- "Open" IE — self-supervised learning of unlexicalized, relation-independent extractors (*e.g.*, Textrunner (Banko et al., 2007)).

Unfortunately, both methods have problems. Supervised approaches require manually-labeled training data for each relation and hence can't scale to handle the thousands of relations encoded in Web text. Open extraction is more scalable, but has lower precision and recall. Furthermore, open extraction doesn't canonicalize relations, so any application using the output must deal with homonymy and synonymy.

A third approach, sometimes refered to as weak supervision, is to heuristically match values from a database to text, thus generating a set of training data for self-supervised learning of relation-specific extractors (Craven and Kumlien, 1999). With the Kylin system (Wu and Weld, 2007) applied this idea to Wikipedia by matching values of an article's infobox[1] attributes to corresponding sentences in the article, and suggested that their approach could extract thousands of relations (Wu et al., 2008). Unfortunately, however, they never tested the idea on more than a dozen relations. Indeed, no one has demonstrated a practical way to extract more than about one hundred relations.

We note that Wikipedia's infobox 'ontology' is a particularly interesting target for extraction. As a by-product of thousands of contributors, it is broad in coverage and growing quickly. Unfortunately, the schemata are surprisingly noisy and most are sparsely populated; challenging conditions for extraction.

This paper presents LUCHS, an autonomous, self-supervised system, which learns 5025 relational extractors — an order of magnitude greater than any previous effort. Like Kylin, LUCHS creates training data by matching Wikipedia attribute values with corresponding sentences, but by itself, this method was insufficient for accurate extraction of most relations. Thus, LUCHS introduces a new technique, *dynamic lexicon features*, which dramatically improves performance when learning from sparse data and that way enables scalability.

1.1 Dynamic Lexicon Features

Figure 1 summarizes the architecture of LUCHS. At the highest level, LUCHS's offline training process resembles that of Kylin. Wikipedia pages

[1] A sizable fraction of Wikipedia articles have associated infoboxes — relational summaries of the key aspects of the subject of the article. For example, the infobox for Alan Turing's Wikipedia page lists the values of 10 attributes, including his birthdate, nationality and doctoral advisor.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 286–295,
Uppsala, Sweden, 11-16 July 2010.

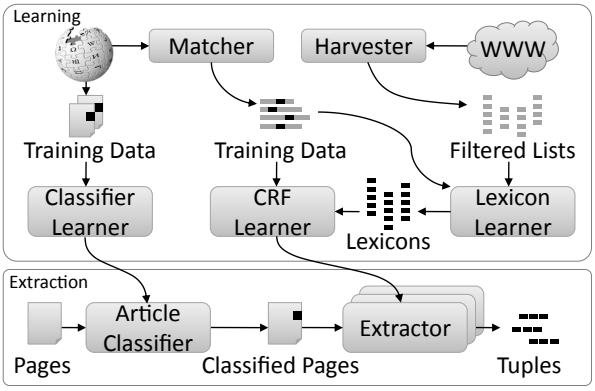


Figure 1: Architecture of LUCHS. In order to handle sparsity in its heuristically-generated training data, LUCHS generates custom lexicon features when learning each relational extractor.

containing infoboxes are used to train a classifier that can predict the appropriate schema for pages missing infoboxes. Additionally, the values of infobox attributes are compared with article sentences to heuristically generate training data. LUCHS's major innovation is a feature-generation process, which starts by harvesting HTML lists from a 5B document Web crawl, discarding 98% to create a set of 49M semantically-relevant lists. When learning an extractor for relation R, LUCHS extracts seed phrases from R's training data and uses a semi-supervised learning algorithm to create several relation-specific lexicons at different points on a precision-recall spectrum. These lexicons form Boolean features which, along with lexical and dependency parser-based features, are used to produce a CRF extractor for each relation — one which performs much better than lexicon-free extraction on sparse training data.

At runtime, LUCHS feeds pages to the article classfier, which predicts which infobox schema is most appropriate for extraction. Then a small set of relation-specific extractors are applied to each sentence, outputting tuples. Our experiments demonstrate a high F1 score, 61%, across the 5025 relational extractors learned.

1.2 Summary

This paper makes several contributions:

- We present LUCHS, a self-supervised IE system capable of learning more than an order of magnitude more relation-specific extractors than previous systems.
- We describe the construction and use of *dynamic lexicon features*, a novel technique, that enables hyper-lexicalized extractors which cope effectively with sparse training data.
- We evaluate the overall end-to-end performance of LUCHS, showing an F1 score of 61% when extracting relations from randomly selected Wikipedia pages.
- We present a comprehensive set of additional experiments, evaluating LUCHS's individual components, measuring the effect of dynamic lexicon features, testing sensitivity to varying amounts of training data, and categorizing the types of relations LUCHS can extract.

2 Heuristic Generation of Training Data

Wikipedia is an ideal starting point for our long-term goal of creating a massive knowledge base of extracted facts for two reasons. First, it is comprehensive, containing a diverse body of content with significant depth. Perhaps more importantly, Wikipedia's structure facilitates self-supervised extraction. Infoboxes are short, manually-created tabular summaries of many articles' key facts — effectively defining a relational schema for that class of entity. Since the same facts are often expressed in both article and ontology, matching values of the ontology to the article can deliver valuable, though noisy, training data.

For example, the Wikipedia article on "Jerry Seinfeld" contains the sentence "Seinfeld was born in Brooklyn, New York." and the article's infobox contains the attribute "birth_place = Brooklyn". By matching the attribute's value "Brooklyn" to the sentence, we can heuristically generate training data for a birth_place extractor. This data is noisy; some attributes will not find matches, while others will find many co-incidental matches.

3 Learning Extractors

We first assume that each Wikipedia infobox attribute corresponds to a unique relation (but see Section 5.6) for which we would like to learn a specific extractor. A major challenge with such an approach is scalability. Running a relation-specific extractor for each of Wikipedia's 34,000 unique infobox attributes on each of Wikipedia's 50 million sentences would require 1.7 trillion extractor executions.

We therefore choose a *hierarchical* approach that combines both article classifiers and relation extractors. For each infobox schema, LUCHS trains a classifier that predicts if an article is likely to contain that schema. Only when an article

is likely to contain a schema, does LUCHS run that schema's relation extractors. To extract infobox attributes from all of Wikipedia, LUCHS now needs orders of magnitude fewer executions.

While this approach does not propagate information from extractors back to article classifiers, experiments confirm that our article classifiers nonetheless deliver accurate results (Section 5.2), reducing the potential benefit of joint inference. In addition, our approach reduces the need for extractors to keep track of the larger context, thus simplifying the extraction problem.

We briefly summarize article classification: We use a linear, multi-class classifier with six kinds of features: words in the article title, words in the first sentence, words in the first sentence which are direct objects to the verb 'to be', article section headers, Wikipedia categories, and their ancestor categories. We use the voted perceptron algorithm (Freund and Schapire, 1999) for training.

More challenging are the attribute extractors, which we wish to be simple, fast, and able to well capture local dependencies. We use a linear-chain conditional random field (CRF) — an undirected graphical model connecting a sequence of input and output random variables, $x = (x_0, \ldots, x_T)$ and $y = (y_0, \ldots, y_T)$ (Lafferty et al., 2001). Input variables are assigned words w. The states of output variables represent discrete labels l, e.g. $\mathsf{Arg}_i\text{-of-}\mathsf{Rel}_j$ and Other. In our case, variables are connected in a chain, following the first-order Markov assumption. We train to maximize conditional likelihood of output variables given an input probability distribution. The CRF models $p(y|x)$ are represented with a log-linear distribution

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

where feature functions, f, encode sufficient statistics of (x, y), T is the length of the sequence, K is the number of feature functions, and λ_k are parameters representing feature weights, which we learn during training. $Z(x)$ is a partition function used to normalize the probabilities to 1. Feature functions allow complex, overlapping global features with lookahead.

Common techniques for learning the weights λ_k include numeric optimization algorithms such as stochastic gradient descent or L-BFGS. In our experiments, we again use the simpler and more efficient voted-perceptron algorithm (Collins, 2002). The linear-chain layout enables efficient interence using the dynamic programming-based Viterbi algorithm (Lafferty et al., 2001).

We evaluate nine kinds of Boolean features:

Words For each input word w we introduce feature $f_w^{\mathbf{w}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t=w]}$.

State Transitions For each transition between output labels l_i, l_j we add feature $f_{l_i,l_j}^{\mathbf{tran}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[y_{t-1}=l_i \wedge y_t=l_j]}$.

Word Contextualization For parameters p and s we add features $f_w^{\mathbf{prev}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[w \in \{x_{t-p}, \ldots, x_{t-1}\}]}$ and $f_w^{\mathbf{sub}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[w \in \{x_{t+1}, \ldots, x_{t+s}\}]}$ which capture a window of words appearing before and after each position t.

Capitalization We add feature $f^{\mathbf{cap}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{is capitalized}]}$.

Digits We add feature $f^{\mathbf{dig}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{is digits}]}$.

Dependencies We set $f^{\mathbf{dep}}(y_{t-1}, y_t, x, t)$ to the lemmatized sequence of words from x_t to the root of the dependency tree, computed using the Stanford parser (Marneffe et al., 2006).

First Sentence We set $f^{\mathbf{fs}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \text{in first sentence of article}]}$.

Gaussians For numeric attributes, we fit a Gaussian (μ, σ) and add feature $f_i^{\mathbf{gau}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[|x_t - \mu| < i\sigma]}$ for parameters i.

Lexicons For non-numeric attributes, and for a lexicon l, *i.e.* a set of related words, we add feature $f_l^{\mathbf{lex}}(y_{t-1}, y_t, x, t) := \mathbb{1}_{[x_t \in l]}$. Lexicons are explained in the following section.

4 Extraction with Lexicons

It is often possible to group words that are likely to be assigned similar labels, even if many of these words do not appear in our training set. The obtained lexicons then provide an elegant way to improve the generalization ability of an extractor, especially when only little training data is available. However, there is a danger of overfitting, which we discuss in Section 4.2.4.

The next section explains how we mine the Web to obtain a large corpus of quality lists. Then Section 4.2 presents our semi-supervised algorithm for learning semantic lexicons from these lists.

4.1 Harvesting Lists from the Web

Domain-independence requires access to an extremely large number of lists, but our tight integration of lexicon acquisition and CRF learning requires that relevant lists be accessed instantaneously. Approaches using search engines or wrappers at query time (Etzioni et al., 2004; Wang and Cohen, 2008) are too slow; we must extract and index lists prior to learning.

We begin with a 5 billion page Web crawl. LUCHS can be combined with any list harvesting technique, but we choose a simple approach, extracting lists defined by HTML `<ul>` or `<ol>` tags. The set of lists obtained in this way is extremely noisy — many lists comprise navigation bars, tag sets, spam links, or a series of long text paragraphs. This is consistent with the observation that less than 2% of Web tables are relational (Cafarella et al., 2008).

We therefore apply a series of filtering steps. We remove lists of only one or two items, lists containing long phrases, and duplicate lists from the same host. After filtering we obtain 49 million lists, containing 56 million unique phrases.

4.2 Semi-Supervised Learning of Lexicons

While training a CRF extractor for a given relation, LUCHS uses its corpus of lists to automatically generate a set of semantic lexicons — *specific to that relation*. The technique proceeds in three steps, which have been engineered to run extremely quickly:

1. Seed phrases are extracted from the labeled training set.
2. A learning algorithm expands the seed phrases into a set of lexicons.
3. The semantic lexicons are added as features to the CRF learning algorithm.

4.2.1 Extracting Seed Phrases

For each training sentence LUCHS first identifies subsequences of labeled words, and for each such labeled subsequence, LUCHS creates one or more seed phrases p. Typically, a set of seeds consists precisely of the labeled subsequences. However, if the labeled subsequences are long and have substructure, *e.g.*, 'San Remo, Italy', our system splits at the separator token, and creates additional seed sets from prefixes and postfixes.

4.2.2 From Seeds to Lexicons

To expand a set of seeds into a lexicon, LUCHS must identify relevant lists in the corpus. Relevancy can be computed by defining a similarity between lists using the vector-space model. Specifically, let $\mathcal{L}$ denote the corpus of lists, and $\mathcal{P}$ be the set of unique phrases from $\mathcal{L}$. Each list $l^0 \in \mathcal{L}$ can be represented as a vector of weighted phrases $p \in \mathcal{P}$ appearing on the list, $l^0 = (l^0_{p_1} l^0_{p_2} \ldots l^0_{p_{|\mathcal{P}|}})$. Following the notion of *inverse document frequency,* a phrase's weight is inversely proportional to the number of lists containing the phrase. Popular phrases which appear on many lists thus receive a small weight, whereas rare phrases are weighted higher:

$$l^0_{p_i} = \frac{1}{|\{l \in \mathcal{L} | p \in l\}|}$$

Unlike the vector space model for documents, we ignore term frequency, since the vast majority of lists in our corpus don't contain duplicates. This vector representation supports the simple cosine definition of *list similarity*, which for lists $l^0, l^1 \in \mathcal{L}$ is defined as

$$sim_{cos} := \frac{l^0 \cdot l^1}{\|l^0\| \|l^1\|}.$$

Intuitively, two lists are similar if they have many overlapping phrases, the phrases are not too common, and the lists don't contain many other phrases. By representing the seed set as another vector, we can find similar lists, hopefully containing related phrases. We then create a semantic lexicon by collecting phrases from a range of related lists.

For example, one lexicon may be created as the union of all phrases on lists that have non-zero similarity to the seed list. Unfortunately, due to the noisy nature of the Web lists such a lexicon may be very large and may contain many irrelevant phrases. We expect that lists with higher similarity are more likely to contain phrases which are related to our seeds; hence, by varying the similarity threshold one may produce lexicons representing different compromises between lexicon precision and recall. Not knowing which lexicon will be most useful to the extractors, LUCHS generates several and lets the extractors learn appropriate weights.

However, since list similarities vary depending on the seeds, fixed thresholds are not an option. If *#similarlists* denotes the number of lists that have non-zero similarity to the seed list and *#lexicons*

the total number of lexicons we want to generate, LUCHS sets lexicon $i \in \{0, \ldots, \#lexicons - 1\}$ to be the union of prases on the

$$\#similarlists^{i/\#lexicons}$$

most similar lists.[2]

4.2.3 Efficiently Creating Lexicons

We create lexicons from lists that are *similar* to our seed vector, so we only consider lists that have at least one phrase in common. Importantly, our index structures allow LUCHS to select the relevant lists efficiently. For each seed, LUCHS retrieves the set of containing lists as a sorted sequence of list identifiers. These sequences are then merged yielding a sequence of list identifiers with associated seed-hit counts. Precomputed list lengths and inverse document frequencies are also retrieved from indices, allowing efficient computation of similarity. The worst case complexity is $O(\log(S)SK)$ where S is the number of seeds and K the maximum number of lists to consider per seed.

4.2.4 Preventing Lexicon Overfitting

Finally, we integrate the acquired semantic lexicons as features into the CRF. Although Section 3 discussed how to use lexicons as CRF features, there are some subtleties. Recall that the lexicons were created from seeds extracted from the training set. If we now train the CRF on the *same* examples that generated the lexicon features, then the CRF will likely overfit, and weight the lexicon features too highly!

Before training, we therefore split the training set into k partitions. For each example in a partition we assign features based on lexicons generated from only the $k-1$ *remaining partitions*. This avoids overfitting and ensures that we will not perform much worse than without lexicon features. When we apply the CRF to our test set, we use the lexicons based on all k partitions. We refer to this technique as *cross-training*.

5 Experiments

We start by evaluating end-to-end performance of LUCHS when applied to Wikipedia text, then analyze the characteristics of its components. Our experiments use the 10/2008 English Wikipedia dump.

[2]For practical reasons, we exclude the case $i =$ #lexicons in our experiments.

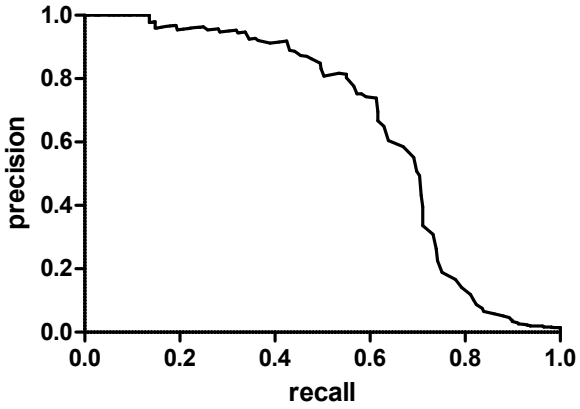


Figure 2: Precision / recall curve for end-to-end system performance on 100 random articles.

5.1 Overall Extraction Performance

To evaluate the end-to-end performance of LUCHS, we test the pipeline which first classifies incoming pages, activating a small set of extractors on the text. To ensure adequate training and test data, we limit ourselves to infobox classes with at least ten instances; there exist 1,583 such classes, together comprising 981,387 articles. We only consider the first ten sentences for each article, and we only consider 5025 attributes.[3] We create a test set by sampling 100 articles randomly; these articles are not used to train article classifiers or extractors. Each test article is then automatically classified, and a random attribute of the predicted schema is selected for extraction. Gold labels for the selected attribute and article are created manually by a human judge and compared to the token-level predictions from the extractors which are trainined on the remaining articles with heuristic matches.

Overall, LUCHS reaches a precision of .55 at a recall of .68, giving an F1-score of .61 (Figure 2). Analyzing the errors in more detail, we find that in 11 of 100 cases an article was incorrectly classified. We note that in at least two of these cases the predicted class could also be considered correct. For example, instead of *Infobox Minor Planet* the extractor predicted *Infobox Planet*.

On five of the selected attributes the extractor failed because the attributes could be considered unlearnable: The flexibility of Wikipedia's infobox system allows contributors to introduce attributes for formatting, for example defining el-

[3]Attributes were selected to have at least 10 heuristic matches, to have 10% of values covered by matches, and 10% of articles with attribute in infobox covered by matches.

ement order. In the future we wish to train LUCHS to ignore this type of attribute.

We also compared the heuristic matches contained in the selected 100 articles to the gold standard: The matches reach a precision of .90 at a recall of .33, giving an F1-score of .48. So while most heuristic matches hit mentions of attribute values, many other mentions go unmatched. Manual analysis shows that these values are often missing from an infobox, are formatted differently, or are inconsistent to what is stated in the article.

So why did the low recall of the heuristic matches not adversely affect recall of our extractors? For most articles, an attribute can be assigned a single unique value. When training an attribute extractor, only articles that contained a heuristic match for that attribute were considered, thus avoiding many cases of unmatched mentions.

Subsequent experiments evaluate the performance of LUCHS components in more detail.

5.2 Article Classification

The first step in LUCHS's run-time pipeline is determining which infobox schemata are most likely to be found in a given article. To test this, we randomly split our 981,387 articles into 4/5 for training and 1/5 for testing, and train a single multi-class classifier. For this experiment, we use the original infobox class of an article as its gold label. We compute the accuracy of the prediction at .92. Since some classes can be considered interchangeable, this number represents a lower bound on performance.

5.3 Factors Affecting Extraction Accuracy

We now evaluate attribute extraction assuming perfect article classification. To keep training time manageable, we sample 100 articles for training and 100 articles for testing[4] for each of 100 random attributes. We again only consider the first ten sentences of each article, and we only consider articles that have heuristic matches with the attribute. We measure F1-score at a token-level, taking the heuristic matches as ground-truth.

We first test the performance of extractors trained using our basic features (Section 3)[5], not including lexicons and Gaussians. We begin using word features and obtain a token-level F1-score of .311 for text and .311 for numeric attributes. Adding any of our additional features improves these scores, but the relative improvements vary: For both text and numeric attributes, contextualization and dependency features deliver the largest improvement. We then iteratively add the feature with largest improvement until no further improvement is observed. We finally obtain an F1-score of .491 for text and .586 for numeric attributes. For text attributes the extractor uses word, contextualization, first sentence, capitalization, and digit features; for numeric attributes the extractor uses word, contextualization, digit, first sentence, and dependency features. We use these extractors as a baseline to evaluate our lexicon and Gaussian features.

Features	F1-Score
Text attributes	
Baseline	.491
Baseline + Lexicons w/o CT	.367
Baseline + Lexicons	.545
Numeric attributes	
Baseline	.586
Baseline + Gaussians w/o CT	.623
Baseline + Gaussians	.627

Table 1: Impact of Lexicon and Gaussian features. Cross-Training (CT) is essential to improve performance.

Varying the size of the training sets affects results: Taking more articles raises the F1-score, but taking more sentences per article reduces it. This is because Wikipedia articles often summarize a topic in the first few paragraphs and later discuss related topics, necessitating reference resolution which we plan to add in future work.

5.4 Lexicon and Gaussian Features

We next study how our distribution features[6] impact the quality of the baseline extractors (Table 1). Without cross-training we observe a reduction in performance, due to overfitting. Cross-training avoids this, and substantially improves results over the baseline. While cross-training is particularly critical for lexicon features, it is less needed for Gaussians where only two parameters, mean and deviation, are fitted to the training set.

The relative improvements depend on the number of available training examples (Table 2). Lexicon and Gaussian features especially benefit extractors for sparse attributes. Here we can also see that the improvements are mainly due to increases in recall.

[4]These numbers are smaller for attributes with less training data available, but the same split is maintained.

[5]For contextualization features we choose $p, s = 5$.

[6]We set the number of lexicon and Gaussian features to 4.

# Train	F1-B	F1-LUCHS	ΔF1	ΔPr	ΔRe
Text attributes					
10	.379	.439	+16%	+10%	+20%
25	.447	.504	+13%	+7%	+20%
100	.491	.545	+11%	+5%	+17%
Numeric attributes					
10	.484	.531	+10%	+4%	+13%
25	.552	.596	+8%	+4%	+10%
100	.586	.627	+7%	+5%	+8%

Table 2: Lexicon and Gaussian features greatly expand F1 score (F1-LUCHS) over the baseline (F1-B), in particular for attributes with few training examples. Gains are mainly due to increased recall.

5.5 Scaling to All of Wikipedia

Finally, we take our best extractors and run them on all 5025 attributes, again assuming perfect article classification and using heuristic matches as gold-standard. Figure 3 shows the distribution of obtained F1 scores. 810 text attributes and 328 numeric attributes reach a score of 0.80 or higher.

The performance depends on the number of available training examples, and that number is governed by a long-tailed distribution. For example, 61% of the attributes in our set have 50 or fewer examples, 36% have 20 or fewer. Interestingly, the number of training examples had a smaller effect on performance than expected. Figure 4 shows the correlation between these variables. Lexicon and Gaussian features enables acceptable performance even for sparse attributes.

Averaging across all attributes we obtain F1 scores of 0.56 and 0.60 for textual and numeric values respectively. We note that these scores assume that all attributes are equally important, weighting rare attributes just like common ones. If we weight scores by the number of attribute instances, we obtain F1 scores of 0.64 (textual) and 0.78 (numeric). In each case, precision is slightly higher than recall.

5.6 Towards an Attribute Ontology

The true promise of relation-specific extractors comes when an ontology ties the system together. By learning a probabilistic model of selectional preferences, one can use joint inference to improve extraction accuracy. One can also answer scientific questions, such as "How many of the learned Wikipedia attributes are distinct?" It is clear that many duplicates exist due to collaborative sloppiness, but semantic similarity is a matter of opinion and an exact answer is impossible.

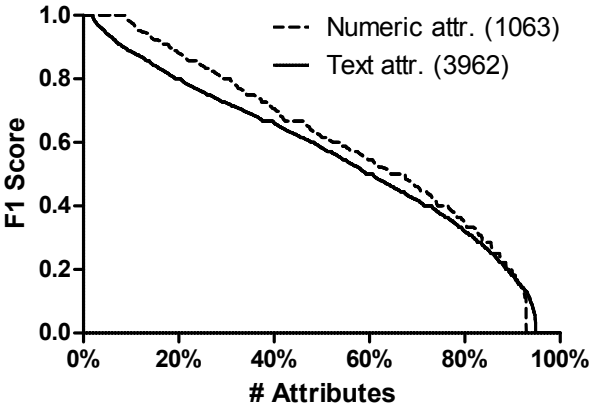


Figure 3: F1 scores among attributes, ranked by score. 810 text attributes (20%) and 328 numeric attributes (31%) had an F1-score of .80 or higher.

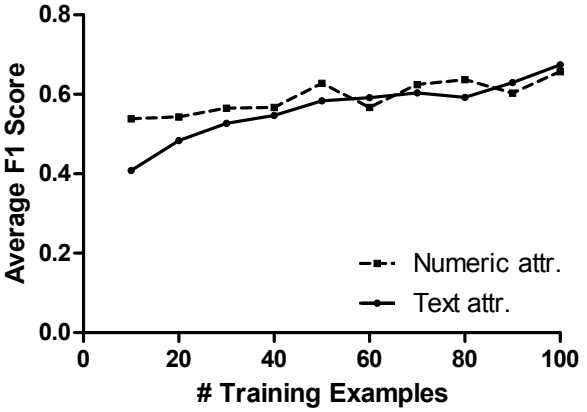


Figure 4: Average F1 score by number of training examples. While more training data helps, even sparse attributes reach acceptable performance.

Nevertheless, we clustered the textual attributes in several ways. First, we cleaned the attribute names heuristically and performed spell check. The "distance" between two attributes was calculated with a combination of edit distance and IR metrics with Wordnet synonyms; then hierarchical agglomerative clustering was performed. We manually assigned names to the clusters and cleaned them, splitting and joining as needed. The result is too crude to be called an ontology, but we continue its elaboration. There are a total of 3962 attributes grouped in about 1282 clusters (not yet counting attributes with numerical values); the largest cluster, *location*, has 115 similar attributes. Figure 5 shows the confusion matrix between attributes in the biggest clusters; the shade of the i, j^{th} pixel indicates the F1 score achieved by training on instances of attribute i and testing on attribute j.

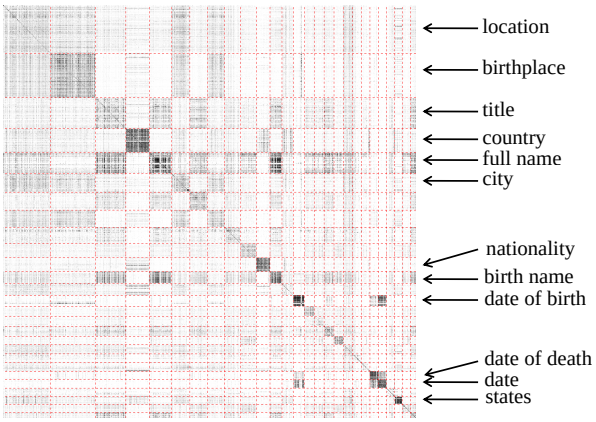


Figure 5: Confusion matrix for extractor accuracy training on one attribute then testing on another. Note the extraction similarity between title and full-name, as well as between dates of birth and death. Space constraints allow us to show only 1000 of LUCHS's 5025 extracted attributes, those in the largest clusters.

6 Related Work

Large-scale extraction A popular approach to IE is supervised learning of relation-specific extractors (Freitag, 1998). Open IE, self-supervised learning of unlexicalized, relation-independent extractors (Banko et al., 2007), is a more scalable approach, but suffers from lower precision and recall, and doesn't canonicalize the relations. A third approach, weak supervision, performs self-supervised learning of relation-specific extractors from noisy training data, heuristically generated by matching database values to text. (Craven and Kumlien, 1999; Hirschman et al., 2002) apply this technique to the biological domain, and (Mintz et al., 2009) apply it to 102 relations from Freebase. LUCHS differs from these approaches in that its "database" – the set of infobox values – itself is noisy, contains many more relations, and has few instances per relation. Whereas the existing approaches focus on *syntactic* extraction patterns, LUCHS focuses on *lexical* information enhanced by dynamic lexicon learning.

Extraction from Wikipedia Wikipedia has become an interesting target for extraction. (Suchanek et al., 2008) build a knowledgebase from Wikipedia's semi-structured data. (Wang et al., 2007) propose a semisupervised positive-only learning technique. Although that extracts from text, its reliance on hyperlinks and other semi-structured data limits extraction. (Wu and Weld, 2007; Wu et al., 2008)'s systems generate training data similar to LUCHS, but were only on a few infobox classes. In contrast, LUCHS shows that the idea scales to more than 5000 relations, but that additional techniques, such as dynamic lexicon learning, are necessary to deal with sparsity.

Extraction with lexicons While lexicons have been commonly used for IE (Cohen and Sarawagi, 2004; Agichtein and Ganti, 2004; Bellare and McCallum, 2007), many approaches assume that lexicons are clean and are supplied by a user *before* training. Other approaches (Talukdar et al., 2006; Miller et al., 2004; Riloff, 1993) learn lexicons automatically from distributional patterns in text. (Wang et al., 2009) learns lexicons from Web lists for query tagging. LUCHS differs from these approaches in that it is not limited to a small set of well-defined relations. Rather than creating large lexicons of common entities, LUCHS attempts to efficiently instantiate a series of lexicons from a small set of seeds to bias extractors of sparse attributes. Crucual to LUCHS's different setting is also the need to avoid overfitting.

Set expansion A large amount of work has looked at automatically generating sets of related items. Starting with a set of seed terms, (Etzioni et al., 2004) extract lists by learning wrappers for Web pages containing those terms. (Wang and Cohen, 2007; Wang and Cohen, 2008) extend the idea, computing term relatedness through a random walk algorithm that takes into account seeds, documents, wrappers and mentions. Other approaches include Bayesian methods (Ghahramani and Heller, 2005) and graph label propagation algorithms (Talukdar et al., 2008; Bengio et al., 2006). The goal of set expansion techniques is to generate high precision sets of related items; hence, these techniques are evaluated based on *lexicon* precision and recall. For LUCHS, which is evaluated based on the quality of an *extractor* using the lexicons, lexicon precision is not important – as long as it does not confuse the extractor.

7 Future Work

We envision a Web-scale machine reading system which simultaneously learns ontologies and extractors, and we believe that LUCHS's approach of leveraging noisy semi-structured information (such as lists or formatting templates) is a key towards this goal. For future work, we plan to enhance LUCHS in two major ways.

First, we note that a big weakness is that the system currently only works for Wikipedia pages.

For example, LUCHS assumes that each page corresponds to exactly one schema and that the subject of relations on a page are the same. Also, LUCHS makes predictions on a token basis, thus sometimes failing to recognize larger segments. To remove these limitations we plan to add a deeper linguistic analysis, making better use of parse and dependency information and including coreference resolution. We also plan to employ relation-independent Open extraction techniques, e.g. as suggested in (Wu and Weld, 2008) (retraining).

Second, we note that LUCHS's performance may benefit substantially from an attribute ontology. As we showed in Section 5.6, LUCHS's current extractors can also greatly facilitate learning a full attribute ontology. We therefore plan to interleave extractor learning and ontology inference, hence jointly learning ontology and extractors.

8 Conclusion

Many researchers are trying to use IE to create large-scale knowledge bases from natural language text on the Web, but existing relation-specific techniques do not scale to the thousands of relations encoded in Web text – while relation-independent techniques suffer from lower precision and recall, and do not canonicalize the relations. This paper shows that – with new techniques – *self-supervised* learning of relation-specific extractors from Wikipedia infoboxes *does* scale.

In particular, we present LUCHS, a self-supervised IE system capable of learning more than an order of magnitude more relation-specific extractors than previous systems. LUCHS uses *dynamic lexicon features* that enable hyper-lexicalized extractors which cope effectively with sparse training data. We show an overall performance of 61% F1 score, and present experiments evaluating LUCHS's individual components.

Datasets generated in this work are available to the community[7].

Acknowledgments

We thank Jesse Davis, Oren Etzioni, Andrey Kolobov, Mausam, Fei Wu, and the anonymous reviewers for helpful comments and suggestions.

This material is based upon work supported by a WRF / TJ Cable Professorship, a gift from Google and by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL).

[7] http://www.cs.washington.edu/ai/iwp

References

Eugene Agichtein and Venkatesh Ganti. 2004. Mining reference tables for automatic text segmentation. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, pages 20–29.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC-2007)*, pages 722–735.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 2670–2676.

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth International Workshop on Information Integration on the Web*.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label propagation and quadratic criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the International Conference on Very Large Databases (VLDB-2008)*, 1(1):538–549.

Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2009a. Coupling semi-supervised learning of categories and relations. In *NAACL HLT 2009 Workskop on Semi-supervised Learning for Natural Language Processing*.

Andrew Carlson, Scott Gaffney, and Flavian Vasile. 2009b. Learning a named entity tagger from gazetteers with the partial perceptron. In *AAAI Spring Symposium on Learning by Reading and Learning to Read*.

William W. Cohen and Sunita Sarawagi. 2004. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, pages 89–98.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB-1999)*, pages 77–86.

Benjamin Van Durme and Marius Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 1243–1248.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, pages 391–398.

Dayne Freitag. 1998. Toward general-purpose learning for information extraction. In *Proceedings of the 17th international conference on Computational linguistics*, pages 404–408. Association for Computational Linguistics.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Zoubin Ghahramani and Katherine A. Heller. 2005. Bayesian sets. In *Neural Information Processing Systems (NIPS-2005)*.

Lynette Hirschman, Alexander A. Morgan, and Alexander S. Yeh. 2002. Rutabaga by any other name: extracting biological names. *Journal of Biomedical Informatics*, 35(4):247–259.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, pages 282–289.

Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC-2006)*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-2004)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *The Annual Meeting of the Association for Computational Linguistics (ACL-2009)*.

Marius Pasca. 2009. Outclassing wikipedia in open-domain information extraction: Weakly-supervised acquisition of attributes over conceptual hierarchies. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2009)*, pages 639–647.

Ellen Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-1993)*, pages 811–816.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from wikipedia and wordnet. *Elsevier Journal of Web Semantics*, 6(3):203–217.

Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. 2009. Sofie: A self-organizing framework for information extraction. In *Proceedings of the 18th International Conference on World Wide Web (WWW-2009)*.

Partha Pratim Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. 2006. A context pattern induction method for named entity extraction. In *The Tenth Conference on Natural Language Learning (CoNLL-X-2006)*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, pages 582–590.

Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM-2007)*, pages 342–350.

Richard C. Wang and William W. Cohen. 2008. Iterative set expansion of named entities using the web. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM-2008)*.

Gang Wang, Yong Yu, and Haiping Zhu. 2007. Pore: Positive-only relation extraction from wikipedia text. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC-2007)*, pages 580–594.

Ye-Yi Wang, Raphael Hoffmann, Xiao Li, and Alex Acero. 2009. Semi-supervised acquisition of semantic classes – from the web and for the web. In *International Conference on Information and Knowledge Management (CIKM-2009)*, pages 37–46.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM-2007)*, pages 41–50.

Fei Wu and Daniel S. Weld. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th International Conference on World Wide Web (WWW-2008)*, pages 635–644.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *The Annual Meeting of the Association for Computational Linguistics (ACL-2010)*.

Fei Wu, Raphael Hoffmann, and Daniel S. Weld. 2008. Information extraction from wikipedia: moving down the long tail. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2008)*, pages 731–739.

Unsupervised Ontology Induction from Text

Hoifung Poon and **Pedro Domingos**
Department of Computer Science & Engineering
University of Washington
hoifung,pedrod@cs.washington.edu

Abstract

Extracting knowledge from unstructured text is a long-standing goal of NLP. Although learning approaches to many of its subtasks have been developed (e.g., parsing, taxonomy induction, information extraction), all end-to-end solutions to date require heavy supervision and/or manual engineering, limiting their scope and scalability. We present OntoUSP, a system that induces and populates a probabilistic ontology using only dependency-parsed text as input. OntoUSP builds on the USP unsupervised semantic parser by jointly forming ISA and IS-PART hierarchies of lambda-form clusters. The ISA hierarchy allows more general knowledge to be learned, and the use of smoothing for parameter estimation. We evaluate OntoUSP by using it to extract a knowledge base from biomedical abstracts and answer questions. OntoUSP improves on the recall of USP by 47% and greatly outperforms previous state-of-the-art approaches.

1 Introduction

Knowledge acquisition has been a major goal of NLP since its early days. We would like computers to be able to read text and express the knowledge it contains in a formal representation, suitable for answering questions and solving problems. However, progress has been difficult. The earliest approaches were manual, but the sheer amount of coding and knowledge engineering needed makes them very costly and limits them to well-circumscribed domains. More recently, machine learning approaches to a number of key subproblems have been developed (e.g., Snow et al. (2006)), but to date there is no sufficiently automatic end-to-end solution. Most saliently, supervised learning requires labeled data, which itself is costly and infeasible for large-scale, open-domain knowledge acquisition.

Ideally, we would like to have an end-to-end unsupervised (or lightly supervised) solution to the problem of knowledge acquisition from text. The TextRunner system (Banko et al., 2007) can extract a large number of ground atoms from the Web using only a small number of seed patterns as guidance, but it is unable to extract non-atomic formulas, and the mass of facts it extracts is unstructured and very noisy. The USP system (Poon and Domingos, 2009) can extract formulas and appears to be fairly robust to noise. However, it is still limited to extractions for which there is substantial evidence in the corpus, and in most corpora most pieces of knowledge are stated only once or a few times, making them very difficult to extract without supervision. Also, the knowledge extracted is simply a large set of formulas without ontological structure, and the latter is essential for compact representation and efficient reasoning (Staab and Studer, 2004).

We propose OntoUSP (Ontological USP), a system that learns an ISA hierarchy over clusters of logical expressions, and populates it by translating sentences to logical form. OntoUSP is encoded in a few formulas of higher-order Markov logic (Domingos and Lowd, 2009), and can be viewed as extending USP with the capability to perform hierarchical (as opposed to flat) clustering. This clustering is then used to perform hierarchical smoothing (a.k.a. shrinkage), greatly increasing the system's capability to generalize from

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 296–305,
Uppsala, Sweden, 11-16 July 2010.

sparse data.

We begin by reviewing the necessary background. We then present the OntoUSP Markov logic network and the inference and learning algorithms used with it. Finally, experiments on a biomedical knowledge acquisition and question answering task show that OntoUSP can greatly outperform USP and previous systems.

2 Background

2.1 Ontology Learning

In general, ontology induction (constructing an ontology) and ontology population (mapping textual expressions to concepts and relations in the ontology) remain difficult open problems (Staab and Studer, 2004). Recently, ontology learning has attracted increasing interest in both NLP and semantic Web communities (Cimiano, 2006; Maedche, 2002), and a number of machine learning approaches have been developed (e.g., Snow et al. (2006), Cimiano (2006), Suchanek et al. (2008,2009), Wu & Weld (2008)). However, they are still limited in several aspects. Most approaches induce and populate a deterministic ontology, which does not capture the inherent uncertainty among the entities and relations. Besides, many of them either bootstrap from heuristic patterns (e.g., Hearst patterns (Hearst, 1992)) or build on existing structured or semi-structured knowledge bases (e.g., WordNet (Fellbaum, 1998) and Wikipedia[1]), thus are limited in coverage. Moreover, they often focus on inducing ontology over individual words rather than arbitrarily large meaning units (e.g., idioms, phrasal verbs, etc.). Most importantly, existing approaches typically separate ontology induction from population and knowledge extraction, and pursue each task in a standalone fashion. While computationally efficient, this is suboptimal. The resulted ontology is disconnected from text and requires additional effort to map between the two (Tsujii, 2004). In addition, this fails to leverage the intimate connections between the three tasks for joint inference and mutual disambiguiation.

Our approach differs from existing ones in two main aspects: we induce a probabilistic ontology from text, and we do so by jointly conducting ontology induction, population, and knowledge extraction. Probabilistic modeling handles uncertainty and noise. A joint approach propagates information among the three tasks, uncovers more implicit information from text, and can potentially work well even in domains not well covered by existing resources like WordNet and Wikipedia. Furthermore, we leverage the ontology for hierarchical smoothing and incorporate this smoothing into the induction process. This facilitates more accurate parameter estimation and better generalization.

Our approach can also leverage existing ontologies and knowledge bases to conduct semi-supervised ontology induction (e.g., by incorporating existing structures as hard constraints or penalizing deviation from them).

2.2 Markov Logic

Combining uncertainty handling and joint inference is the hallmark of the emerging field of statistical relational learning (a.k.a. structured prediction), where a plethora of approaches have been developed (Getoor and Taskar, 2007; Bakir et al., 2007). In this paper, we use Markov logic (Domingos and Lowd, 2009), which is the leading unifying framework, but other approaches can be used as well. Markov logic is a probabilistic extension of first-order logic and can compactly specify probability distributions over complex relational domains. It has been successfully applied to unsupervised learning for various NLP tasks such as coreference resolution (Poon and Domingos, 2008) and semantic parsing (Poon and Domingos, 2009). A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by the log-linear model $P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith formula, and n_i is the number of satisfied groundings.

2.3 Unsupervised Semantic Parsing

Semantic parsing aims to obtain a complete canonical meaning representation for input sentences. It can be viewed as a structured prediction problem, where a semantic parse is formed by partitioning the input sentence (or a syntactic analysis such as a dependency tree) into meaning units and assigning each unit to the logical form representing an entity or relation (Figure 1). In effect, a semantic

[1] `http://www.wikipedia.org`

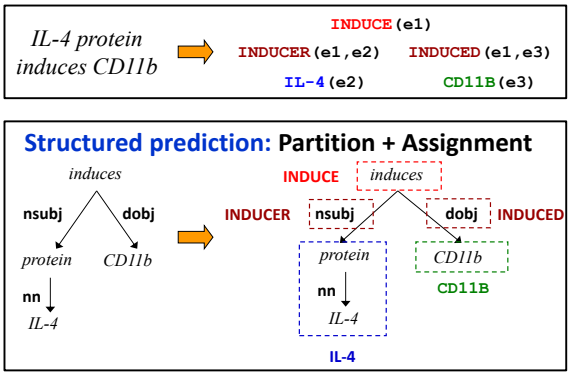


Figure 1: An example of semantic parsing. Top: semantic parsing converts an input sentence into logical form in Davidsonian semantics. Bottom: a semantic parse consists of a partition of the dependency tree and an assignment of its parts.

parser extracts knowledge from input text and converts them into logical form (the semantic parse), which can then be used in logical and probabilistic inference and support end tasks such as question answering.

A major challenge to semantic parsing is syntactic and lexical variations of the same meaning, which abound in natural languages. For example, the fact that IL-4 protein induces CD11b can be expressed in a variety of ways, such as, "Interleukin-4 enhances the expression of CD11b", "CD11b is upregulated by IL-4", etc. Past approaches either manually construct a grammar or require example sentences with meaning annotation, and do not scale beyond restricted domains.

Recently, we developed the USP system (Poon and Domingos, 2009), the first unsupervised approach for semantic parsing.[2] USP inputs dependency trees of sentences and first transforms them into quasi-logical forms (QLFs) by converting each node to a unary atom and each dependency edge to a binary atom (e.g., the node for "induces" becomes $\texttt{induces}(\texttt{e}_1)$ and the subject dependency becomes $\texttt{nsubj}(\texttt{e}_1, \texttt{e}_2)$, where $\texttt{e}_\texttt{i}$'s are Skolem constants indexed by the nodes.).[3] For each sentence, a semantic parse comprises of a partition of its QLF into subexpressions, each of which has a naturally corresponding lambda

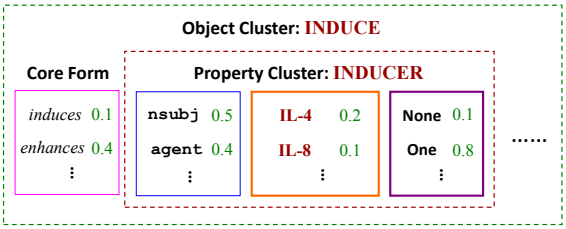


Figure 2: An example of object/property clusters: INDUCE contains the core-form property cluster and others, such as the agent argument INDUCER.

form,[4] and an assignment of each subexpression to a lambda-form cluster.

The lambda-form clusters naturally form an IS-PART hierarchy (Figure 2). An *object cluster* corresponds to semantic concepts or relations such as INDUCE, and contains a variable number of *property clusters*. A special property cluster of *core forms* maintains a distribution over variations in lambda forms for expressing this concept or relation. Other property clusters correspond to modifiers or arguments such as INDUCER (the agent argument of INDUCE), each of which in turn contains three subclusters of property values: the argument-object subcluster maintains a distribution over object clusters that may occur in this argument (e.g., IL − 4), the argument-form subcluster maintains a distribution over lambda forms that corresponds to syntactic variations for this argument (e.g., nsubj in active voice and agent in passive voice), and the argument-number subcluster maintains a distribution over total numbers of this argument that may occur in a sentence (e.g., zero if the argument is not mentioned).

Effectively, USP simultaneously discovers the lambda-form clusters and an IS-PART hierarchy among them. It does so by recursively combining subexpressions that are composed with or by similar subexpressions. The partition breaks a sentence into subexpressions that are meaning units, and the clustering abstracts away syntactic and lexical variations for the same meaning. This novel form of relational clustering is governed by a joint probability distribution $P(T, L)$ defined in higher-order[5] Markov logic, where T are the input dependency trees, and L the semantic parses. The

[2] In this paper, we use a slightly different formulation of USP and its MLN to facilitate the exposition of OntoUSP.

[3] We call these QLFs because they are not true logical form (the ambiguities are not yet resolved). This is related to but not identical with the definition in Alshawi (1990).

[4] The lambda form is derived by replacing every Skolem constant $\texttt{e}_\texttt{i}$ that does not appear in any unary atom in the subexpression with a lambda variable $\texttt{x}_\texttt{i}$ that is uniquely indexed by the corresponding node i. For example, the lambda form for $\texttt{nsubj}(\texttt{e}_1, \texttt{e}_2)$ is $\lambda\texttt{x}_1\lambda\texttt{x}_2.\texttt{nsubj}(\texttt{x}_1, \texttt{x}_2)$.

[5] Variables can range over arbitrary lambda forms.

main predicates are:

$\texttt{e} \in \texttt{c}$: expression e is assigned to cluster c;

$\texttt{SubExpr}(\texttt{s}, \texttt{e})$: s is a subexpression of e;

$\texttt{HasValue}(\texttt{s}, \texttt{v})$: s is of value v;

$\texttt{IsPart}(\texttt{c}, \texttt{i}, \texttt{p})$: p is the property cluster in object cluster c uniquely indexed by i.

In USP, property clusters in different object clusters use distinct index i's. As we will see later, in OntoUSP, property clusters with ISA relation share the same index i, which corresponds to a generic semantic frame such as agent and patient.

The probability model of USP can be captured by two formulas:

$$\begin{aligned} &\texttt{x} \in +\texttt{p} \wedge \texttt{HasValue}(\texttt{x}, +\texttt{v}) \\ &\texttt{e} \in \texttt{c} \wedge \texttt{SubExpr}(\texttt{x}, \texttt{e}) \wedge \texttt{x} \in \texttt{p} \\ &\quad \Rightarrow \exists^1 \texttt{i}.\texttt{IsPart}(\texttt{c}, \texttt{i}, \texttt{p}). \end{aligned}$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the variables with a plus sign. The first formula is the core of the model and represents the mixture of property values given the cluster. The second formula ensures that a property cluster must be a part in the corresponding object cluster; it is a hard constraint, as signified by the period at the end.

To encourage clustering, USP imposes an exponential prior over the number of parameters.

To parse a new sentence, USP starts by partitioning the QLF into atomic forms, and then hill-climbs on the probability using a search operator based on lambda reduction until it finds the maximum a posteriori (MAP) parse. During learning, USP starts with clusters of atomic forms, maintains the optimal semantic parses according to current parameters, and hill-climbs on the log-likelihood of observed QLFs using two search operators:

$\texttt{MERGE}(\texttt{c}_1, \texttt{c}_2)$ merges clusters $\texttt{c}_1, \texttt{c}_2$ into a larger cluster c by merging the core-form clusters and argument clusters of $\texttt{c}_1, \texttt{c}_2$, respectively. E.g., $\texttt{c}_1 = \{\text{“}\textit{induce}\text{”}\}$, $\texttt{c}_2 = \{\text{“}\textit{enhance}\text{”}\}$, and $\texttt{c} = \{\text{“}\textit{induce}\text{”}, \text{“}\textit{enhance}\text{”}\}$.

$\texttt{COMPOSE}(\texttt{c}_1, \texttt{c}_2)$ creates a new lambda-form cluster c formed by composing the lambda forms in $\texttt{c}_1, \texttt{c}_2$ into larger ones. E.g., $\texttt{c}_1 = \{\text{“}\textit{amino}\text{”}\}$, $\texttt{c}_2 = \{\text{“}\textit{acid}\text{”}\}$, and $\texttt{c} = \{\text{“}\textit{amino acid}\text{”}\}$.

Each time, USP executes the highest-scored operator and reparses affected sentences using the new parameters. The output contains the optimal lambda-form clusters and parameters, as well as the MAP semantic parses of input sentences.

3 Unsupervised Ontology Induction with Markov Logic

A major limitation of USP is that it either merges two object clusters into one, or leaves them separate. This is suboptimal, because different object clusters may still possess substantial commonalities. Modeling these can help extract more general knowledge and answer many more questions. The best way to capture such commonalities is by forming an ISA hierarchy among the clusters. For example, INDUCE and INHIBIT are both subconcepts of REGULATE. Learning these ISA relations helps answer questions like "What regulates CD11b?", when the text states that "IL-4 induces CD11b" or "AP-1 suppresses CD11b".

For parameter learning, this is also undesirable. Without the hierarchical structure, each cluster estimates its parameters solely based on its own observations, which can be extremely sparse. The better solution is to leverage the hierarchical structure for smoothing (a.k.a. shrinkage (McCallum et al., 1998; Gelman and Hill, 2006)). For example, if we learn that "super-induce" is a verb and that in general verbs have active and passive voices, then even though "super-induce" only shows up once in the corpus as in "AP-1 is super-induced by IL-4", by smoothing we can still infer that this probably means the same as "IL-4 super-induces AP-1", which in turn helps answer questions like "What super-induces AP-1".

OntoUSP overcomes the limitations of USP by replacing the flat clustering process with a hierarchical clustering one, and learns an ISA hierarchy of lambda-form clusters in addition to the IS-PART one. The output of OntoUSP consists of an ontology, a semantic parser, and the MAP parses. In effect, OntoUSP conducts ontology induction, population, and knowledge extraction in a single integrated process. Specifically, given clusters $\texttt{c}_1, \texttt{c}_2$, in addition to merge vs. separate, OntoUSP evaluates a third option called *abstraction*, in which a new object cluster c is created, and ISA links are added from $\texttt{c}_\texttt{i}$ to c; the argument clusters in c are formed by merging that of $\texttt{c}_\texttt{i}$'s.

In the remainder of the section, we describe the

details of OntoUSP. We start by presenting the OntoUSP MLN. We then describe our inference algorithm and how to parse a new sentence using OntoUSP. Finally, we describe the learning algorithm and how OntoUSP induces the ontology while learning the semantic parser.

3.1 The OntoUSP MLN

The OntoUSP MLN can be obtained by modifying the USP MLN with three simple changes. First, we introduce a new predicate $\texttt{IsA}(\texttt{c}_1, \texttt{c}_2)$, which is true if cluster $\texttt{c}_1$ is a subconcept of $\texttt{c}_2$. For convenience, we stipulate that IsA is reflexive (i.e., $\texttt{IsA}(\texttt{c}, \texttt{c})$ is true for any c). Second, we add two formulas to the MLN:

$$\texttt{IsA}(\texttt{c}_1, \texttt{c}_2) \wedge \texttt{IsA}(\texttt{c}_2, \texttt{c}_3) \Rightarrow \texttt{IsA}(\texttt{c}_1, \texttt{c}_3).$$
$$\texttt{IsPart}(\texttt{c}_1, \texttt{i}_1, \texttt{p}_1) \wedge \texttt{IsPart}(\texttt{c}_2, \texttt{i}_2, \texttt{p}_2) \wedge \texttt{IsA}(\texttt{c}_1, \texttt{c}_2) \Rightarrow (\texttt{i}_1 = \texttt{i}_2 \Leftrightarrow \texttt{IsA}(\texttt{p}_1, \texttt{p}_2)).$$

The first formula simply enforces the transitivity of ISA relation. The second formula states that if the ISA relation holds for a pair of object clusters, it also holds between their corresponding property clusters. Both are hard constraints. Third, we introduce hierarchical smoothing into the model by replacing the USP mixture formula

$$\texttt{x} \in +\texttt{p} \wedge \texttt{HasValue}(\texttt{x}, +\texttt{v})$$

with a new formula

$$\texttt{ISA}(\texttt{p}_1, +\texttt{p}_2) \ \wedge \ \texttt{x} \in \texttt{p}_1 \ \wedge \texttt{HasValue}(\texttt{x}, +\texttt{v})$$

Intuitively, for each $\texttt{p}_2$, the weight corresponds to the delta in log-probability of v comparing to the prediction according to all ancestors of $\texttt{p}_2$. The effect of this change is that now the value v of a subexpression x is not solely determined by its property cluster $\texttt{p}_1$, but is also smoothed by statistics of all $\texttt{p}_2$ that are super clusters of $\texttt{p}_1$.

Shrinkage takes place via interaction among the weights of the ISA mixture formula. In particular, if the weights for some property cluster p are all zero, it means that values in p are completely predicted by p's ancestors. In effect, p is backed off to its parent.

3.2 Inference

Given the dependency tree T of a sentence, the conditional probability of a semantic parse L is given by $Pr(L|T) \propto \exp\left(\sum_i w_i n_i(T, L)\right)$. The MAP semantic parse is simply $\arg\max_L \sum_i w_i n_i(T, L)$. Directly enumerating all L's is intractable. OntoUSP uses the same inference algorithm as USP by hill-climbing on the probability of L; in each step, OntoUSP evaluates the alternative semantic parses that can be formed by lambda-reducing a current subexpression with one of its arguments. The only difference is that OntoUSP uses a different MLN and so the probabilities and resulting semantic parses may be different. Algorithm 1 gives pseudo-code for OntoUSP's inference algorithm.

Algorithm 1 OntoUSP-Parse(*MLN, T*)

```
Initialize semantic parse L with individual atoms in the QLF of T
repeat
  for all subexpressions e in L do
    Evaluate all semantic parses that are lambda-reducible from e
  end for
  L ← the new semantic parse with the highest gain in probability
until none of these improve the probability
return L
```

3.3 Learning

OntoUSP uses the same learning objective as USP, i.e., to find parameters θ that maximizes the log-likelihood of observing the dependency trees T, summing out the unobserved semantic parses L:

$$\begin{aligned} L_\theta(T) &= \log P_\theta(L) \\ &= \log \textstyle\sum_L P_\theta(T, L) \end{aligned}$$

However, the learning problem in OntoUSP is distinct in two important aspects. First, OntoUSP learns in addition an ISA hierarchy among the lambda-form clusters. Second and more importantly, OntoUSP leverages this hierarchy during learning to smooth the parameter estimation of individual clusters, as embodied by the new ISA mixture formula in the OntoUSP MLN.

OntoUSP faces several new challenges unseen in previous hierarchical-smoothing approaches. The ISA hierarchy in OntoUSP is not known in advance, but needs to be learned as well. Similarly, OntoUSP has no known examples of populated facts and rules in the ontology, but has to infer that in the same joint learning process. Finally, OntoUSP does not start from well-formed structured input like relational tuples, but rather directly from raw text. In sum, OntoUSP tackles a

Algorithm 2 OntoUSP-Learn(*MLN, T's*)

```
Initialize with a flat ontology, along with clusters and semantic parses
Merge clusters with the same core form
Agenda ← ∅
repeat
  for all candidate operations O do
    Score O by log-likelihood improvement
    if score is above a threshold then
      Add O to agenda
    end if
  end for
  Execute the highest scoring operation O* in the agenda
  Regenerate MAP parses for affected trees and update agenda and candidate operations
until agenda is empty
return the learned ontology and MLN, and the semantic parses
```

very hard problem with exceedingly little aid from user supervision.

To combat these challenges, OntoUSP adopts a novel form of hierarchical smoothing by integrating it with the search process for identifying the hierarchy. Algorithm 2 gives pseudocode for OntoUSP's learning algorithm. Like USP, OntoUSP approximates the sum over all semantic parses with the most probable parse, and searches for both θ and the MAP semantic parses L that maximize $P_\theta(T, L)$. In addition to MERGE and COMPOSE, OntoUSP uses a new operator $\texttt{ABSTRACT}(\texttt{c}_1, \texttt{c}_2)$, which does the following:

1. Create an *abstract* cluster $\texttt{c}$;
2. Create ISA links from $\texttt{c}_1, \texttt{c}_2$ to $\texttt{c}$;
3. Align property clusters of $\texttt{c}_1$ and $\texttt{c}_2$; for each aligned pair $\texttt{p}_1$ and $\texttt{p}_2$, either merge them into a single property cluster, or create an *abstract* property cluster $\texttt{p}$ in $\texttt{c}$ and create ISA links from $\texttt{p}_\texttt{i}$ to $\texttt{p}$, so as to maximize log-likelihood.

Intuitively, $\texttt{c}$ corresponds to a more abstract concept that summarizes similar properties in $\texttt{c}_\texttt{i}$'s.

To add a child cluster $\texttt{c}_2$ to an existing abstract cluster $\texttt{c}_1$, OntoUSP also uses an operator $\texttt{ADDCHILD}(\texttt{c}_1, \texttt{c}_2)$ that does the following:

1. Create an ISA link from $\texttt{c}_2$ to $\texttt{c}_1$;
2. For each property cluster of $\texttt{c}_2$, maximize the log-likelihood by doing one of the following: merge it with a property cluster in an existing child of $\texttt{c}_1$; create ISA link from it to an abstract property cluster in $\texttt{c}$; leave it unchanged.

For efficiency, in both operators, the best option is chosen greedily for each property cluster in $\texttt{c}_2$, in descending order of cluster size.

Notice that once an abstract cluster is created, it could be merged with an existing cluster using MERGE. Thus with the new operators, OntoUSP is capable of inducing any ISA hierarchy among abstract and existing clusters. (Of course, the ISA hierarchy it actually induces depends on the data.)

Learning the shrinkage weights has been approached in a variety of ways; examples include EM and cross-validation (McCallum et al., 1998), hierarchical Bayesian methods (Gelman and Hill, 2006), and maximum entropy with L_1 priors (Dudik et al., 2007). The past methods either only learn parameters with one or two levels (e.g., in hierarchical Bayes), or requires significant amount of computation (e.g., in EM and in L_1-regularized maxent), while also typically assuming a given hierarchy. In contrast, OntoUSP has to both induce the hierarchy and populate it, with potentially many levels in the induced hierarchy, starting from raw text with little user supervision.

Therefore, OntoUSP simplifies the weight learning problem by adopting standard m-estimation for smoothing. Namely, the weights for cluster $\texttt{c}$ are set by counting its observations plus m fractional samples from its parent distribution. When $\texttt{c}$ has few observations, its unreliable statistics can be significantly augmented via the smoothing by its parent (and in turn to a gradually smaller degree by its ancestors). m is a hyperparameter that can be used to trade off bias towards statistics for parent vs oneself.

OntoUSP also needs to balance between two conflicting aspects during learning. On one hand, it should encourage creating abstract clusters to summarize intrinsic commonalities among the children. On the other hand, this needs to be heavily regularized to avoid mistaking noise for the signal. OntoUSP does this by a combination of priors and thresholding. To encourage the induction of higher-level nodes and inheritance, OntoUSP imposes an exponential prior β on the number of *parameter slots*. Each slot corresponds to a distinct property value. A child cluster inherits its parent's slots (and thus avoids the penalty on them). On-

toUSP also stipulates that, in an ABSTRACT operation, a new property cluster can be created either as a *concrete* cluster with full parameterization, or as an *abstract* cluster that merely serves for smoothing purposes. To discourage overproposing clusters and ISA links, OntoUSP imposes a large exponential prior γ on the number of concrete clusters created by ABSTRACT. For *abstract* cluster, it sets a cut-off t_p and only allows storing a probability value no less than t_p. Like USP, it also rejects MERGE and COMPOSE operations that improve log-likelihood by less than t_o. These priors and cut-off values can be tuned to control the granularity of the induced ontology and clusters.

Concretely, given semantic parses L, OntoUSP computes the optimal parameters and evaluates the regularized log-likelihood as follows. Let $w_{p_2,v}$ denote the weight of the ISA mixture formula $\texttt{ISA}(\texttt{p}_1, +\texttt{p}_2) \wedge \texttt{x} \in \texttt{p}_1 \wedge \texttt{HasValue}(\texttt{x}, +\texttt{v})$. For convenience, for each pair of property cluster c and value v, OntoUSP instead computes and stores $w'_{\texttt{c,v}} = \sum_{\texttt{ISA}(\texttt{c},\texttt{a})} w_{\texttt{a,v}}$, which sums over all weights for c and its ancestors. (Thus $w_{\texttt{c,v}} = w'_{\texttt{c,v}} - w'_{\texttt{p,v}}$, where p is the parent of c.) Like USP, OntoUSP imposes local normalization constraints that enable closed-form estimation of the optimal parameters and likelihood. Specifically, using m-estimation, the optimal $w'_{\texttt{c,v}}$ is $\log((m \cdot e^{w'_{\texttt{p,v}}} + n_{\texttt{c,v}})/(m + n_{\texttt{c}}))$, where p is the parent of c and n is the count. The log-likelihood is $\sum_{\texttt{c,v}} w'_{\texttt{c,v}} \cdot n_{\texttt{c,v}}$, which is then augmented by the priors.

4 Experiments

4.1 Methodology

Evaluating unsupervised ontology induction is difficult, because there is no gold ontology for comparison. Moreover, our ultimate goal is to aid knowledge acquisition, rather than just inducing an ontology for its own sake. Therefore, we used the same methodology and dataset as the USP paper to evaluate OntoUSP on its capability in knowledge acquisition. Specifically, we applied OntoUSP to extract knowledge from the GENIA dataset (Kim et al., 2003) and answer questions, and we evaluated it on the number of extracted answers and accuracy. GENIA contains 1999 PubMed abstracts.[6] The question set contains 2000 questions which were created by sampling verbs and entities according to their frequencies in GENIA. Sample questions include "What regulates MIP-1alpha?", "What does anti-STAT 1 inhibit?". These simple question types were used to focus the evaluation on the knowledge extraction aspect, rather than engineering for handling special question types and/or reasoning.

[6] http://www-tsujii.is.s.u-tokyo-.ac.jp/GENIA/home/wiki.cgi.

4.2 Systems

OntoUSP is the first unsupervised approach that synergistically conducts ontology induction, population, and knowledge extraction. The system closest in aim and capability is USP. We thus compared OntoUSP with USP and all other systems evaluated in the USP paper (Poon and Domingos, 2009). Below is a brief description of the systems. (For more details, see Poon & Domingos (2009).)

Keyword is a baseline system based on keyword matching. It directly matches the question substring containing the verb and the available argument with the input text, ignoring case and morphology. Given a match, two ways to derive the answer were considered: KW simply returns the rest of sentence on the other side of the verb, whereas KW-SYN is informed by syntax and extracts the answer from the subject or object of the verb, depending on the question (if the expected argument is absent, the sentence is ignored).

TextRunner (Banko et al., 2007) is the state-of-the-art system for open-domain information extraction. It inputs text and outputs relational triples in the form (R, A_1, A_2), where R is the relation string, and A_1, A_2 the argument strings. To answer questions, each triple-question pair is considered in turn by first matching their relation strings, and then the available argument strings. If both match, the remaining argument string in the triple is returned as an answer. Results were reported when exact match is used (TR-EXACT), or when the triple strings may contain the question ones as substrings (TR-SUB).

RESOLVER (Yates and Etzioni, 2009) inputs TextRunner triples and collectively resolves coreferent relation and argument strings. To answer questions, the only difference from TextRunner is that a question string can match any string in its cluster. As in TextRunner, results were reported for both exact match (RS-EXACT) and substring (RS-SUB).

DIRT (Lin and Pantel, 2001) resolves binary rela-

Table 1: Comparison of question answering results on the GENIA dataset. Results for systems other than OntoUSP are from Poon & Domingos (2009).

	# Total	# Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP	334	295	88%
OntoUSP	**480**	**435**	**91%**

tions by inputting a dependency path that signifies the relation and returns a set of similar paths. To use DIRT in question answering, it was queried to obtain similar paths for the relation of the question, which were then used to match sentences.

USP (Poon and Domingos, 2009) parses the input text using the Stanford dependency parser (Klein and Manning, 2003; de Marneffe et al., 2006), learns an MLN for semantic parsing from the dependency trees, and outputs this MLN and the MAP semantic parses of the input sentences. These MAP parses formed the knowledge base (KB). To answer questions, USP first parses the questions (with the question slot replaced by a dummy word), and then matches the question parse to parses in the KB by testing subsumption.

OntoUSP uses a similar procedure as USP for extracting knowledge and answering questions, except for two changes. First, USP's learning and parsing algorithms are replaced with OntoUSP-Learn and OntoUSP-Parse, respectively. Second, when OntoUSP matches a question to its KB, it not only considers the lambda-form cluster of the question relation, but also all its sub-clusters.[7]

4.3 Results

Table 1 shows the results comparing OntoUSP with other systems. While USP already greatly outperformed other systems in both precision and recall, OntoUSP further substantially improved on the recall of USP, without any loss in precision. In particular, OntoUSP extracted 140 more correct answers than USP, for a gain of 47% in absolute

[7] Additional details are available at `http://alchemy.cs.washington.edu/papers/poon10`.

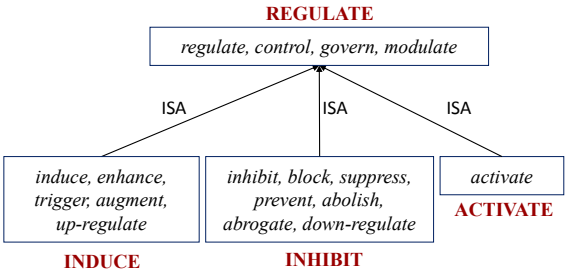


Figure 3: A fragment of the induced ISA hierarchy, showing the core forms for each cluster (the cluster labels are added by the authors for illustration purpose).

recall. Compared to TextRunner (TR-SUB), OntoUSP gained on precision by 38 points and extracted more than five times of correct answers.

Manual inspection shows that the induced ISA hierarchy is the key for the recall gain. Like USP, OntoUSP discovered the following clusters (in core forms) that represent some of the core concepts in biomedical research:

{regulate, control, govern, modulate}

{induce, enhance, trigger, augment, up-regulate}

{inhibit, block, suppress, prevent, abolish, abrogate, down-regulate}

However, USP formed these as separate clusters, whereas OntoUSP in addition induces ISA relations from the `INDUCE` and `INHIBIT` clusters to the `REGULATE` cluster (Figure 3). This allows OntoUSP to answer many more questions that are asked about general regulation events, even though the text states them with specific regulation directions like "induce" or "inhibit". Below is an example question-answer pair output by OntoUSP; neither USP nor any other system were able to extract the necessary knowledge.

Q: What does IL-2 control?

A: The DEX-mediated IkappaBalpha induction.

Sentence: Interestingly, the DEX-mediated IkappaBalpha induction was completely inhibited by IL-2, but not IL-4, in Th1 cells, while the reverse profile was seen in Th2 cells.

OntoUSP also discovered other interesting commonalities among the clusters. For example, both USP and OntoUSP formed a singleton cluster with core form "activate". Although this cluster may appear similar to the `INDUCE` cluster, the data in GENIA does not support merging the two. However, OntoUSP discovered that

the ACTIVATE cluster, while not completely resolvent with INDUCE, shared very similar distributions in their agent arguments. In fact, they are so similar that OntoUSP merges them into a single property cluster. It found that the patient arguments of INDUCE and INHIBIT are very similar and merged them. In turn, OntoUSP formed ISA links from these three object clusters to REGULATE, as well as among their property clusters. Intuitively, this makes sense. The positive- and negative-regulation events, as signified by INDUCE and INHIBIT, often target similar object entities or processes. However, their agents tend to differ since in one case they are inducers, and in the other they are inhibitors. On the other hand, ACTIVATE and INDUCE share similar agents since they both signify positive regulation. However, "activate" tends to be used more often when the patient argument is a concrete entity (e.g., cells, genes, proteins), whereas "induce" and others are also used with processes and events (e.g., expressions, inhibition, pathways).

USP was able to resolve common syntactic differences such as active vs. passive voice. However, it does so on the basis of individual verbs, and there is no generalization beyond their clusters. OntoUSP, on the other hand, formed a high-level cluster with two abstract property clusters, corresponding to general agent argument and patient argument. The active-passive alternation is captured in these clusters, and is inherited by all descendant clusters, including many rare verbs like "super-induce" which only occur once in GENIA and for which there is no way that USP could have learned about their active-passive alternations. This illustrates the importance of discovering ISA relations and performing hierarchical smoothing.

4.4 Discussion

OntoUSP is a first step towards joint ontology induction and knowledge extraction. The experimental results demonstrate the promise in this direction. However, we also notice some limitations in the current system. While OntoUSP induced meaningful ISA relations among relation clusters like REGULATE, INDUCE, etc., it was less successful in inducing ISA relations among entity clusters such as specific genes and proteins. This is probably due to the fact that our model only considers local features such as the parent and arguments. A relation is often manifested as verbs and has several arguments, whereas an entity typically appears as an argument of others and has few arguments of its own. As a result, in average, there is less information available for entities than relations. Presumably, we can address this limitation by modeling longer-ranged dependencies such as grandparents, siblings, etc. This is straightforward to do in Markov logic.

OntoUSP also uses a rather elaborate scheme for regularization. We hypothesize that this can be much simplified and improved by adopting a principled framework such as Dudik et al. (2007).

5 Conclusion

This paper introduced OntoUSP, the first unsupervised end-to-end system for ontology induction and knowledge extraction from text. OntoUSP builds on the USP semantic parser by adding the capability to form hierarchical clusterings of logical expressions, linked by ISA relations, and using them for hierarchical smoothing. OntoUSP greatly outperformed USP and other state-of-the-art systems in a biomedical knowledge acquisition task.

Directions for future work include: exploiting the ontological structure for principled handling of antonyms and (more generally) expressions with opposite meanings; developing and testing alternate methods for hierarchical modeling in OntoUSP; scaling up learning and inference to larger corpora; investigating the theoretical properties of OntoUSP's learning approach and generalizing it to other tasks; answering questions that require inference over multiple extractions; etc.

6 Acknowledgements

We give warm thanks to the anonymous reviewers for their comments. This research was partly funded by ARO grant W911NF-08-1-0242, AFRL contract FA8750-09-C-0181, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, HR0011-06-C-0025, HR0011-07-C-0060 and NBCH-D030010, NSF grants IIS-0534881 and IIS-0803481, and ONR grant N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, DARPA, NSF, ONR, or the United States Government.

References

Hiyan Alshawi. 1990. Resolving quasi logical forms. *Computational Linguistics*, 16:133–144.

G. Bakir, T. Hofmann, B. B. Schölkopf, A. Smola, B. Taskar,

S. Vishwanathan, and (eds.). 2007. *Predicting Structured Data*. MIT Press, Cambridge, MA.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India. AAAI Press.

Philipp Cimiano. 2006. *Ontology learning and population from text*. Springer.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Miroslav Dudik, David Blei, and Robert Schapire. 2007. Hierarchical maximum entropy density estimation. In *Proceedings of the Twenty Fourth International Conference on Machine Learning*.

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

Andrew Gelman and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:180–82.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the Forty First Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, CA. ACM Press.

Alexander Maedche. 2002. *Ontology learning for the semantic Web*. Kluwer Academic Publishers, Boston, Massachusetts.

Andrew McCallum, Ronald Rosenfeld, Tom Mitchell, and Andrew Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore. ACL.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL 2006*.

S. Staab and R. Studer. 2004. *Handbook on ontologies*. Springer.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago - a large ontology from Wikipedia and WordNet. *Journal of Web Semantics*.

Fabian Suchanek, Mauro Sozio, and Gerhard Weikum. 2009. Sofie: A self-organizing framework for information extraction. In *Proceedings of the Eighteenth International Conference on World Wide Web*.

Jun-ichi Tsujii. 2004. Thesaurus or logical ontology, which do we need for mining text? In *Proceedings of the Language Resources and Evaluation Conference*.

Fei Wu and Daniel S. Weld. 2008. Automatically refining the wikipedia infobox ontology. In *Proceedings of the Seventeenth International Conference on World Wide Web*, Beijing, China.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Exploring Syntactic Structural Features for Sub-Tree Alignment using Bilingual Tree Kernels

Jun Sun[1,2] **Min Zhang**[1] **Chew Lim Tan**[2]
[1] Institute for Infocomm Research [2]School of Computing, National University of Singapore
sunjun@comp.nus.edu.sg mzhang@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

We propose Bilingual Tree Kernels (BTKs) to capture the structural similarities across a pair of syntactic translational equivalences and apply BTKs to sub-tree alignment along with some plain features. Our study reveals that the structural features embedded in a bilingual parse tree pair are very effective for sub-tree alignment and the bilingual tree kernels can well capture such features. The experimental results show that our approach achieves a significant improvement on both gold standard tree bank and automatically parsed tree pairs against a heuristic similarity based method. We further apply the sub-tree alignment in machine translation with two methods. It is suggested that the sub-tree alignment benefits both phrase and syntax based systems by relaxing the constraint of the word alignment.

1 Introduction

Syntax based Statistical Machine Translation (SMT) systems allow the translation process to be more grammatically performed, which provides decent reordering capability. However, most of the syntax based systems construct the syntactic translation rules based on *word alignment*, which not only suffers from the pipeline errors, but also fails to effectively utilize the syntactic structural features. To address those deficiencies, Tinsley et al. (2007) attempt to directly capture the syntactic translational equivalences by automatically conducting sub-tree alignment, which can be defined as follows:

A sub-tree alignment process pairs up sub-tree pairs across bilingual parse trees whose contexts are semantically translational equivalent. According to Tinsley et al. (2007), a sub-tree aligned parse tree pair follows the following criteria:

(i) a node can only be linked once;

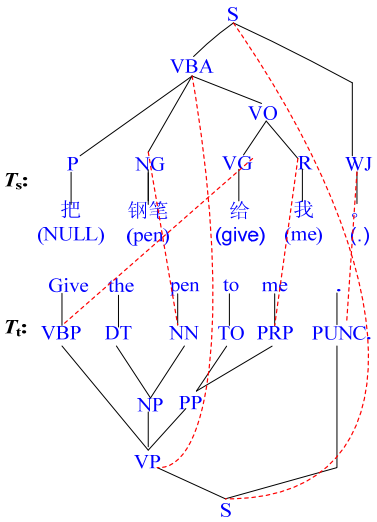


Figure 1: Sub-tree alignment as referred to Node alignment

(ii) descendants of a source linked node may only link to descendants of its target linked counterpart;

(iii) ancestors of a source linked node may only link to ancestors of its target linked counterpart.

By sub-tree alignment, translational equivalent sub-tree pairs are coupled as aligned counterparts. Each pair consists of both the lexical constituents and their maximum tree structures generated over the lexical sequences in the original parse trees. Due to the 1-to-1 mapping between sub-trees and tree nodes, sub-tree alignment can also be considered as node alignment by conducting multiple links across the internal nodes as shown in Fig. 1.

Previous studies conduct sub-tree alignments by either using a rule based method or conducting some similarity measurement only based on lexical features. Groves et al. (2004) conduct sub-tree alignment by using some heuristic rules, lack of extensibility and generality. Tinsley et al. (2007)

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 306–315,
Uppsala, Sweden, 11-16 July 2010.

and Imamura (2001) propose some score functions based on the lexical similarity and co-occurrence. These works fail to utilize the structural features, rendering the syntactic rich task of sub-tree alignment less convincing and attractive. This may be due to the fact that the syntactic structures in a parse tree pair are hard to describe using plain features. In addition, explicitly utilizing syntactic tree fragments results in exponentially high dimensional feature vectors, which is hard to compute. Alternatively, convolution parse tree kernels (Collins and Duffy, 2001), which implicitly explore the tree structure information, have been successfully applied in many NLP tasks, such as Semantic parsing (Moschitti, 2004) and Relation Extraction (Zhang et al. 2006). However, all those studies are carried out in monolingual tasks. In multilingual tasks such as machine translation, tree kernels are seldom applied.

In this paper, we propose Bilingual Tree Kernels (BTKs) to model the bilingual translational equivalences, in our case, to conduct sub-tree alignment. This is motivated by the decent effectiveness of tree kernels in expressing the similarity between tree structures. We propose two kinds of BTKs named dependent Bilingual Tree Kernel (dBTK), which takes the sub-tree pair as a whole and independent Bilingual Tree Kernel (iBTK), which individually models the source and the target sub-trees. Both kernels can be utilized within different feature spaces using various representations of the sub-structures.

Along with BTKs, various lexical and syntactic structural features are proposed to capture the correspondence between bilingual sub-trees using a polynomial kernel. We then attempt to combine the polynomial kernel and BTKs to construct a composite kernel. The sub-tree alignment task is considered as a binary classification problem. We employ a kernel based classifier with the composite kernel to classify each candidate of sub-tree pair as *aligned* or *unaligned*. Then a greedy search algorithm is performed according to the three criteria of sub-tree alignment within the space of candidates classified as *aligned*.

We evaluate the sub-tree alignment on both the gold standard tree bank and an automatically parsed corpus. Experimental results show that the proposed BTKs benefit sub-tree alignment on both corpora, along with the lexical features and the plain structural features. Further experiments in machine translation also suggest that the obtained sub-tree alignment can improve the performance of both phrase and syntax based SMT systems.

2 Bilingual Tree Kernels

In this section, we propose the two BTKs and study their capability and complexity in modeling the bilingual structural similarity. Before elaborating the concepts of BTKs, we first illustrate some notations to facilitate further understanding.

Each sub-tree pair $(S \cdot T)$ can be explicitly decomposed into multiple sub-structures which belong to the given sub-structure spaces. $\mathcal{S}_I = \{s_1, \dots, s_i, \dots, s_I\}$ refers to the *source* tree sub-structure space; while $\mathcal{T}_J = \{t_1, \dots, t_j, \dots, t_J\}$ refers to the *target* sub-structure space. A sub-structure pair (s_i, t_j) refers to an element in the set of the Cartesian product of the two sub-structure spaces: $(s_i, t_j) \in \mathcal{S}_I \times \mathcal{T}_J$.

2.1 Independent Bilingual Tree Kernel (iBTK)

Given the sub-structure spaces $\mathcal{S}_I$ and $\mathcal{T}_J$, we construct two vectors using the integer counts of the source and target sub-structures:

$$\phi(S) = \left(\#(s_1), \dots, \#(s_k), \dots, \#(s_{|\mathcal{S}_I|})\right)$$
$$\phi(T) = \left(\#(t_1), \dots, \#(t_k), \dots, \#(t_{|\mathcal{T}_J|})\right)$$

where $\#(s_k)$ and $\#(t_k)$ are the numbers of occurrences of the sub-structures s_k and t_k. In order to compute the dot product of the feature vectors in the exponentially high dimensional feature space, we introduce the tree kernel functions as follows:

$$\mathcal{K}_{iBTK}(S \cdot T, S' \cdot T') = \mathcal{K}(S, S') + \mathcal{K}(T, T')$$

The iBTK is defined as a composite kernel consisting of a source tree kernel and a target tree kernel which measures the source and the target structural similarity respectively. Therefore, the composite kernel can be computed using the ordinary monolingual tree kernels (Collins and Duffy, 2001).

$$\begin{aligned}&\mathcal{K}(S, S') = <\phi(S), \phi(S')> \\ &= \sum_{i=1}^{|\mathcal{S}_I|}\left(\sum_{n_s \in N_S} I_i(n_s) \cdot \sum_{n'_s \in N'_S} I_i(n'_s)\right) \\ &= \sum_{n_s \in N_S}\sum_{n'_s \in N'_S} \Delta(n_s, n'_s)\end{aligned}$$

where N_S and N'_S refer to the node sets of the source sub-tree S and S' respectvely. $I_i(n_s)$ is an indicator function which equals to 1 iff the sub-structure s_i is rooted at the node n_s and 0 otherwise. $\Delta(n_s, n'_s) = \sum_{i=1}^{|\mathcal{S}_I|}\left(I_i(n_s) \cdot I_i(n'_s)\right)$ is the number of identical sub-structures rooted at n_s and n'_s. Then we compute the $\Delta(n_s, n'_s)$ function as follows:

(1) If the production rule at n_s and n'_s are different, $\Delta(n_s, n'_s) = 0$;
(2)else if both $n_s$and $n'_s$are POS tags, $\Delta(n_s, n'_s) = \lambda$;
(3)else, $\Delta(n_s, n'_s) = \lambda \prod_{l=1}^{nc(n_s)} \left(1 + \Delta\left(c(n_s, l), c(n'_s, l)\right)\right)$.

where $nc(n_s)$ is the child number of n_s, $c(n_s, l)$ is the l^{th} child of n_s, λ is the decay factor used to make the kernel value less variable with respect to the number of sub-structures.

Similarly, we can decompose the target kernel as $\mathcal{K}(T, T') = \sum_{n_t \in N_T} \sum_{n'_t \in N'_T} \Delta(n_t, n'_t)$ and run the algorithm above as well.

The disadvantage of the iBTK is that it fails to capture the correspondence across the sub-structure pairs. However, the composite style of constructing the iBTK helps keep the computational complexity comparable to the monolingual tree kernel, which is $O(|N_S| \cdot |N'_S| + |N_T| \cdot |N'_T|)$.

2.2 Dependent Bilingual Tree Kernel (dBTK)

The iBTK explores the structural similarity of the source and the target sub-trees respectively. As an alternative, we further define a kernel to capture the relationship across the counterparts without increasing the computational complexity. As a result, we propose the dependent Bilingual Tree kernel (dBTK) to jointly evaluate the similarity across sub-tree pairs by enlarging the feature space to the Cartesian product of the two sub-structure sets.

A dBTK takes the source and the target sub-structure pair as a whole and recursively calculate over the joint sub-structures of the given sub-tree pair. We define the dBTK as follows:

Given the sub-structure space $\mathcal{S}_I \times \mathcal{T}_J$, we construct a vector using the integer counts of the sub-structure pairs to represent a sub-tree pair:

$$\phi(S \cdot T) = \begin{pmatrix} \#(s_1, t_1), \dots, \#(s_1, t_{|\mathcal{T}_J|}), \#(s_2, t_1), \\ \dots, \#(s_{|\mathcal{S}_I|}, t_1), \dots, \#(s_{|\mathcal{S}_I|}, t_{|\mathcal{T}_J|}) \end{pmatrix}$$

where $\#(s_i, t_j)$ is the number of occurrences of the sub-structure pair (s_i, t_j).

$$\begin{aligned}
&\mathcal{K}_{dBTK}(S \cdot T, S' \cdot T') \\
&= < \phi(S \cdot T), \phi(S' \cdot T') > \\
&= \sum_{k=1}^{|\mathcal{S}_I \times \mathcal{T}_J|} \begin{pmatrix} \sum_{n_s \in N_S} \sum_{n_t \in N_T} I_k(n_s, n_t) \cdot \\ \sum_{n'_s \in N'_S} \sum_{n'_t \in N'_T} I_k(n'_s, n'_t) \end{pmatrix} \\
&= \sum_{n_s \in N_S} \sum_{n_t \in N_T} \sum_{n'_s \in N'_S} \sum_{n'_t \in N'_T} \Delta \begin{pmatrix} (n_s, n_t), \\ (n'_s, n'_t) \end{pmatrix} \quad (1) \\
&= \sum_{n_s \in N_S} \sum_{n_t \in N_T} \sum_{n'_s \in N'_S} \sum_{n'_t \in N'_T} \begin{pmatrix} \Delta(n_s, n'_s) \cdot \\ \Delta(n_t, n'_t) \end{pmatrix} \quad (2) \\
&= \sum_{n_s \in N_S} \sum_{n'_s \in N'_S} \Delta(n_s, n'_s) \sum_{n_t \in N_T} \sum_{n'_t \in N'_T} \Delta(n_t, n'_t) \\
&= \mathcal{K}(S, S') \cdot \mathcal{K}(T, T')
\end{aligned}$$

It is infeasible to explicitly compute the kernel function by expressing the sub-trees as feature vectors. In order to achieve convenient computation, we deduce the kernel function as the above.

The deduction from (1) to (2) is derived according to the fact that the number of identical sub-structure pairs rooted in the node pairs (n_s, n_t) and (n'_s, n'_t) equals to the product of the respective counts. As a result, the dBTK can be evaluated as a product of two monolingual tree kernels. Here we verify the correctness of the kernel by directly constructing the feature space for the inner product. Alternatively, Cristianini and Shawe-Taylor (2000) prove the positive semi-definite characteristic of the tensor product of two kernels. The decomposition benefits the efficient computation to use the algorithm for the monolingual tree kernel in Section 2.1.

The computational complexity of the dBTK is still $O(|N_S| \cdot |N'_S| + |N_T| \cdot |N'_T|)$.

3 Sub-structure Spaces for BTKs

The syntactic translational equivalences under BTKs are evaluated with respective to the sub-structures factorized from the candidate sub-tree pairs. In this section, we propose different sub-structures to facilitate the measurement of syntactic similarity for sub-tree alignment. Since the proposed BTKs can be computed by individually evaluating the source and target monolingual tree kernels, the definition of the sub-structure can be simplified to base only on monolingual sub-trees.

3.1 Subset Tree

Motivated from Collins and Duffy (2002) in monolingual tree kernels, the Subset Tree (SST) can be employed as sub-structures. An SST is any sub-graph, which includes more than one non-terminal node, with the constraint that the entire rule productions are included. Fig. 2 shows an example of the SSTs decomposed from the source sub-tree rooted at VP*.

3.2 Root directed Subset Tree

Monolingual Tree kernels achieve decent performance using the SSTs due to the rich exploration of syntactic information. However, the sub-tree alignment task requires strong capability of discriminating the sub-trees with their roots across adjacent generations, because those candidates share many identical SSTs. As illustrated in Fig 2, the source sub-tree rooted at VP*, which should be aligned to the target sub-tree rooted at NP*, may be likely aligned to the sub-tree rooted at PP*,

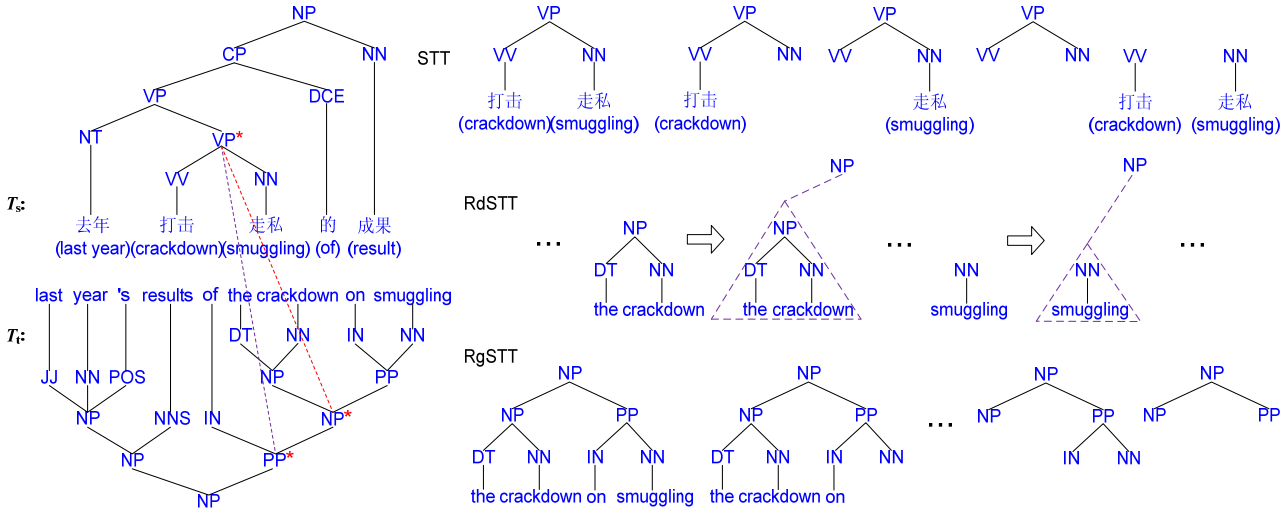


Figure 2: Illustration of SST, RdSST and RgSST

which shares quite a similar context with NP*. It is also easy to show that the latter shares all the SSTs that the former obtains. In consequence, the values of the SST based kernel function are quite similar between the candidate sub-tree pair rooted at (VP*,NP*) and (VP*,PP*).

In order to effectively differentiate the candidates like the above, we propose the Root directed Subset Tree (RdSST) by encapsulating each SST with the root of the given sub-tree. As shown in Fig 2, a sub-structure is considered identical to the given examples, when the SST is identical and the root tag of the given sub-tree is NP. As a result, the kernel function in Section 2.1 is re-defined as:

$$\mathcal{K}(S,S') = \sum_{n_s \in N_S} \sum_{n'_s \in N'_S} \Delta(n_s, n'_s) I(r_s, r'_s) = I(r_s, r'_s) \sum_{n_s \in N_S} \sum_{n'_s \in N'_S} \Delta(n_s, n'_s)$$

where r_s and r'_s are the root nodes of the sub-tree S and S' respectively. The indicator function $I(r_s, r'_s)$ equals to 1 if r_s and r'_s are identical, and 0 otherwise. Although defined for individual SST, the indicator function can be evaluated outside the summation, without increasing the computational complexity of the kernel function.

3.3 Root generated Subset Tree

Some grammatical tags (NP/VP) may have identical tags as their parents or children which may make RdSST less effective. Consequently, we step further to propose the sub-structure of Root generated Subset Tree (RgSST). An RgSST requires the root node of the given sub-tree to be part of the sub-structure. In other words, all sub-structures should be generated from the root of the given sub-tree as presented in Fig. 2. Therefore the kernel function can be simplified to only capture the sub-structure rooted at the root of the sub-tree.

$$\mathcal{K}(S,S') = \Delta(r_s, r'_s)$$

where r_s and r'_s are the root nodes of the sub-tree S and S' respectively. The time complexity is reduced to $O(|N_S| + |N'_S| + |N_T| + |N'_T|)$.

3.4 Root only

More aggressively, we can simplify the kernel to only measure the common root node without considering the complex tree structures. Therefore the kernel function is simplified to be a binary function with time complexity $O(1)$.

$$\mathcal{K}(S,S') = I(r_s, r'_s)$$

4 Plain features

Besides BTKs, we introduce various plain lexical features and structural features which can be expressed as feature functions. The lexical features with directions are defined as conditional feature functions based on the conditional lexical translation probabilities. The plain syntactic structural features can deal with the structural divergence of bilingual parse trees in a more general perspective.

4.1 Lexical and Word Alignment Features

In this section, we define seven lexical features to measure semantic similarity of a given sub-tree pair.

Internal Lexical Features: We define two lexical features with respective to the internal span of the sub-tree pair.

$$\phi_1(S|T) = \left(\prod_{v \in in(T)} \sum_{u \in in(S)} P(u|v)\right)^{\frac{1}{|in(T)|}}$$

$$\phi_2(T|S) = \left(\prod_{u\in in(S)} \sum_{v\in in(T)} P(v|u)\right)^{\frac{1}{|in(S)|}}$$

where $P(v|u)$ refers to the lexical translation probability from the source word u to the target word v within the sub-tree spans, while $P(u|v)$ refers to that from target to source; $in(S)$ refers to the word set for the internal span of the source sub-tree S, while $in(T)$ refers to that of the target sub-tree T.

Internal-External Lexical Features: These features are motivated by the fact that lexical translation probabilities within the translational equivalence tend to be high, and that of the non-equivalent counterparts tend to be low.

$$\phi_3(S|T) = \left(\prod_{v\in in(T)} \sum_{u\in out(S)} P(u|v)\right)^{\frac{1}{|in(T)|}}$$

$$\phi_4(T|S) = \left(\prod_{u\in in(S)} \sum_{v\in out(T)} P(v|u)\right)^{\frac{1}{|in(S)|}}$$

where $out(S)$ refers to the word set for the external span of the source sub-tree S, while $out(T)$ refers to that of the target sub-tree T.

Internal Word Alignment Features: The word alignment links account much for the co-occurrence of the aligned terms. We define the internal word alignment features as follows:

$$\phi_5(S,T) = \frac{\sum_{v\in in(T)} \sum_{u\in in(S)} \delta(u,v) \cdot \left(P(u|v) \cdot P(v|u)\right)^{\frac{1}{2}}}{(|in(S)| \cdot |in(T)|)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

The binary function $\delta(u,v)$ is employed to trigger the computation only when a word aligned link exists for the two words (u,v) within the sub-tree span.

Internal-External Word Alignment Features: Similar to the lexical features, we also introduce the internal-external word alignment features as follows:

$$\phi_6(S,T) = \frac{\sum_{v\in in(T)} \sum_{u\in out(S)} \delta(u,v) \cdot \left(P(u|v) \cdot P(v|u)\right)^{\frac{1}{2}}}{(|out(S)| \cdot |in(T)|)^{\frac{1}{2}}}$$

$$\phi_7(S,T) = \frac{\sum_{v\in out(T)} \sum_{u\in in(S)} \delta(u,v) \cdot \left(P(u|v) \cdot P(v|u)\right)^{\frac{1}{2}}}{(|in(S)| \cdot |out(T)|)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

4.2 Online Structural Features

In addition to the lexical correspondence, we also capture the structural divergence by introducing the following tree structural features.

Span difference: Translational equivalent sub-tree pairs tend to share similar length of spans. Thus the model will penalize the candidate sub-tree pairs with largely different length of spans.

$$\varphi_1(S,T) = \left|\frac{|in(S)|}{|in(\mathbf{S})|} - \frac{|in(T)|}{|in(\mathbf{T})|}\right|$$

S and **T** refer to the entire source and target parse trees respectively. Therefore, $|in(\mathbf{S})|$ and $|in(\mathbf{T})|$ are the respective span length of the parse tree used for normalization.

Number of Descendants: Similarly, the number of the root's descendants of the aligned sub-trees should also correspond.

$$\varphi_2(S,T) = \left|\frac{|R(S)|}{|R(\mathbf{S})|} - \frac{|R(T)|}{|R(\mathbf{T})|}\right|$$

where $R(.)$ refers to the descendant set of the root to a sub-tree.

Tree Depth difference: Intuitively, translational equivalent sub-tree pairs tend to have similar depth from the root of the parse tree. We allow the model to penalize the candidate sub-tree pairs with quite different distance of path from the root of the parse tree to the root of the sub-tree.

$$\varphi_3(S,T) = \left|\frac{Depth(S)}{Height(\mathbf{S})} - \frac{Depth(T)}{Height(\mathbf{T})}\right|$$

5 Alignment Model

Given feature spaces defined in the last two sections, we propose a 2-phase sub-tree alignment model as follows:

In the 1st phase, a kernel based classifier, SVM in our study, is employed to classify each candidate sub-tree pair as *aligned* or *unaligned*. The feature vector of the classifier is computed using a composite kernel:

$$\mathcal{K}(S\cdot T, S'\cdot T') =$$

$$\theta_0 \widehat{\mathcal{K}}_p(S\cdot T, S'\cdot T') + \sum_{\mathrm{i}=1}^{\mathrm{K}} \theta_{\mathrm{i}} \widehat{\mathcal{K}}_{BTK}^{i}(S\cdot T, S'\cdot T')$$

$\widehat{\mathcal{K}}_p(\cdot,\cdot)$ is the normalized form of the polynomial kennel $\mathcal{K}_p(\cdot,\cdot)$, which is a polynomial kernel with the degree of 2, utilizing the plain features. $\widehat{\mathcal{K}}_{BTK}^{i}(\cdot,\cdot)$ is the normalized form of the BTK $\mathcal{K}_{BTK}^{i}(\cdot,\cdot)$, exploring the corresponding sub-structure space. The composite kernel can be constructed using the polynomial kernel for plain features and various BTKs for tree structure by linear combination with coefficient θ_{i}, where $\sum_{\mathrm{i}=0}^{\mathrm{K}} \theta_{\mathrm{i}} = 1$.

In the 2nd phase, we adopt a greedy search with respect to the alignment probabilities. Since SVM is a large margin based discriminative classifier rather than a probabilistic model, we introduce a sigmoid function to convert the distance against the hyperplane to a posterior alignment probability as follows:

$$P(a_+|S,T) = \frac{1}{1+e^{-D_+}}$$

$$P(a_-|S,T) = \frac{1}{1+e^{-D_-}}$$

where D_+ is the distance for the instances classified as *aligned* and D_- is that for the *unaligned*. We use $P(a_+|S,T)$ as the confidence to conduct the *sure* links for those classified as *aligned*. On this perspective, the alignment probability is suitable as a searching metric. The search space is reduced to that of the candidates classified as *aligned* after the 1st phase.

6 Experiments on Sub-Tree Alignments

In order to evaluate the effectiveness of the alignment model and its capability in the applications requiring syntactic translational equivalences, we employ two corpora to carry out the sub-tree alignment evaluation. The first is HIT gold standard English Chinese parallel tree bank referred as HIT corpus[1]. The other is the automatically parsed bilingual tree pairs selected from FBIS corpus (allowing minor parsing errors) with human annotated sub-tree alignment.

6.1 Data preparation

HIT corpus, which is collected from English learning text books in China as well as example sentences in dictionaries, is used for the gold standard corpus evaluation. The word segmentation, tokenization and parse-tree in the corpus are manually constructed or checked. The corpus is constructed with manually annotated sub-tree alignment. The annotation strictly reserves the semantic equivalence of the aligned sub-tree pair. Only *sure* links are conducted in the internal node level, without considering *possible* links adopted in word alignment. A different annotation criterion of the Chinese parse tree, designed by the annotator, is employed. Compared with the widely used Penn TreeBank annotation, the new criterion utilizes some different grammar tags and is able to effectively describe some rare language phenomena in Chinese. The annotator still uses Penn TreeBank annotation on the English side. The statistics of HIT corpus used in our experiment is shown in Table 1. We use 5000 sentences for experiment and divide them into three parts, with 3k for training, 1k for testing and 1k for tuning the parameters of kernels and thresholds of pruning the negative instances.

[1]HIT corpus is designed and constructed by HIT-MITLAB. http://mitlab.hit.edu.cn/index.php/resources.html .

	Chinese	English
# of Sentence pair	5000	
Avg. Sentence Length	12.93	12.92
Avg. # of sub-tree	21.40	23.58
Avg. # of alignment	11.60	

Table 1. Corpus Statistics for HIT corpus

Most linguistically motivated syntax based SMT systems require an automatic parser to perform the rule induction. Thus, it is important to evaluate the sub-tree alignment on the automatically parsed corpus with parsing errors. In addition, HIT corpus is not applicable for MT experiment due to the problems of domain divergence, annotation discrepancy (Chinese parse tree employs a different grammar from Penn Treebank annotations) and degree of tolerance for parsing errors.

Due to the above issues, we annotate a new data set to apply the sub-tree alignment in machine translation. We randomly select 300 bilingual sentence pairs from the Chinese-English FBIS corpus with the length ≤ 30 in both the source and target sides. The selected plain sentence pairs are further parsed by Stanford parser (Klein and Manning, 2003) on both the English and Chinese sides. We manually annotate the sub-tree alignment for the automatically parsed tree pairs according to the definition in Section 1. To be fully consistent with the definition, we strictly reserve the semantic equivalence for the aligned sub-trees to keep a high precision. In other words, we do not conduct any doubtful links. The corpus is further divided into 200 aligned tree pairs for training and 100 for testing as shown in Table 2.

	Chinese	English
# of Sentence pair	300	
Avg. Sentence Length	16.94	20.81
Avg. # of sub-tree	28.97	34.39
Avg. # of alignment	17.07	

Table 2. Statistics of FBIS selected Corpus

6.2 Baseline approach

We implement the work in Tinsley et al. (2007) as our baseline methodology.

Given a tree pair $< \mathbf{S}, \mathbf{T} >$, the baseline approach first takes all the links between the sub-tree pairs as alignment hypotheses, i.e., the Cartesian product of the two sub-tree sets:

$$\{S_1, \dots, S_i, \dots, S_I\} \times \{T_1, \dots, T_j, \dots, T_J\}$$

By using the lexical translation probabilities, each hypothesis is assigned an alignment score. All hypotheses with zero score are pruned out.

Feature Space	P	R	F
Lex	61.62	58.33	59.93
Lex +Online Str	70.08	69.02	69.54
Plain +dBTK-STT	80.36	78.08	79.20
Plain +dBTK-RdSTT	87.52	74.13	80.27
Plain +dBTK-RgSTT	88.54	70.18	78.30
Plain +dBTK-Root	81.05	84.38	**82.68**
Plain +iBTK-STT	81.57	73.51	77.33
Plain +iBTK-RdSTT	82.27	77.85	80.00
Plain +iBTK-RgSTT	82.92	78.77	**80.80**
Plain +iBTK-Root	76.37	76.81	76.59
Plain +dBTK-Root +iBTK-RgSTT	85.53	85.12	**85.32**
Baseline	64.14	66.99	65.53

Table 3. Structure feature contribution for HIT test set
*Plain= Lex +Online Str

Feature Space	P	R	F
Lex	73.48	71.66	72.56
Lex +Online Str	77.02	73.63	75.28
Plain +dBTK-STT	81.44	74.42	77.77
Plain +dBTK-RdSTT	81.40	69.29	74.86
Plain +dBTK-RgSTT	81.90	67.32	73.90
Plain +dBTK-Root	78.60	80.90	**79.73**
Plain +iBTK-STT	82.94	79.44	81.15
Plain +iBTK-RdSTT	83.14	80	**81.54**
Plain +iBTK-RgSTT	83.09	79.72	81.37
Plain +iBTK-Root	78.61	79.49	79.05
Plain +dBTK-Root +iBTK-RdSTT	82.70	82.70	**82.70**
Baseline	70.48	78.70	74.36

Table 4. Structure feature contribution for FBIS test set

Then the algorithm iteratively selects the link of the sub-tree pairs with the maximum score as a *sure* link, and blocks all hypotheses that contradict with this link and itself, until no non-blocked hypotheses remain.

The baseline system uses many heuristics in searching the optimal solutions with alternative score functions. Heuristic *skip1* skips the tied hypotheses with the same score, until it finds the highest-scoring hypothesis with no competitors of the same score. Heuristic *skip2* deals with the same problem. Initially, it skips over the tied hypotheses. When a hypothesis sub-tree pair (S_i, T_j) without any competitor of the same score is found, where neither S_i nor T_j has been skipped over, the hypothesis is chosen as a *sure* link. Heuristic *span1* postpones the selection of the hypotheses on the POS level. Since the highest-scoring hypotheses tend to appear on the leaf nodes, it may introduce ambiguity when conducting the alignment for a POS node whose child word appears twice in a sentence.

The baseline method proposes two score functions based on the lexical translation probability. They also compute the score function by splitting the tree into the internal and external components.

Tinsley et al. (2007) adopt the lexical translation probabilities dumped by GIZA++ (Och and Ney, 2003) to compute the span based scores for each pair of sub-trees. Although all of their heuristics combinations are re-implemented in our study, we only present the best result among them with the highest Recall and F-value as our baseline, denoted as *skip2_s1_span1*[2].

[2] s1 denotes score function 1 in Tinsley et al. (2007), skip2_s1_span1 denotes the utilization of heuristics skip2 and span1 while using score function 1

6.3 Experimental settings

We use SVM with binary classes as the classifier. In case of the implementation, we modify the Tree Kernel tool (Moschitti, 2004) and SVMLight (Joachims, 1999). The coefficient θ_i for the composite kernel are tuned with respect to F-measure (**F**) on the development set of HIT corpus. We empirically set C=2.4 for SVM and use $\alpha = 0.23$, the default parameter $\lambda = 0.4$ for BTKs.

Since the negative training instances largely overwhelm the positive instances, we prune the negative instances using the thresholds according to the lexical feature functions $(\phi_1, \phi_2, \phi_3, \phi_4)$ and online structural feature functions $(\varphi_1, \varphi_2, \varphi_3)$. Those thresholds are also tuned on the development set of HIT corpus with respect to F-measure.

To learn the lexical and word alignment features for both the proposed model and the baseline method, we train GIZA++ on the entire FBIS bilingual corpus (240k). The evaluation is conducted by means of Precision (**P**), Recall (**R**) and F-measure (**F**).

6.4 Experimental results

In Tables 3 and 4, we incrementally enlarge the feature spaces in certain order for both corpora and examine the feature contribution to the alignment results. In detail, the iBTKs and dBTKs are firstly combined with the polynomial kernel for plain features individually, then the best iBTK and dBTK are chosen to construct a more complex composite kernel along with the polynomial kernel for both corpora. The experimental results show that:

- All the settings with structural features of the proposed approach achieve better performance than the baseline method. This is because the

baseline only assesses semantic similarity using the lexical features. The improvement suggests that the proposed framework with syntactic structural features is more effective in modeling the bilingual syntactic correspondence.

- By introducing BTKs to construct a composite kernel, the performance in both corpora is significantly improved against only using the polynomial kernel for plain features. This suggests that the structural features captured by BTKs are quite useful for the sub-tree alignment task. We also try to use BTKs alone without the polynomial kernel for plain features; however, the performance is rather low. This suggests that the structure correspondence cannot be used to measure the semantically equivalent tree structures alone, since the same syntactic structure tends to be reused in the same parse tree and lose the ability of disambiguation to some extent. In other words, to capture the semantic similarity, structure features requires lexical features to cooperate.
- After comparing iBTKs with the corresponding dBTKs, we find that for *FBIS* corpus, iBTK greatly outperforms dBTK in any feature space except the Root space. However, when it comes the *HIT* corpus, the gaps between the corresponding iBTKs and dBTKs are much closer, while on the Root space, dBTK outperforms iBTK to a large amount. This finding can be explained by the relationship between the amount of training data and the high dimensional feature space. Since dBTKs are constructed in a joint manner which obtains a much larger high dimensional feature space than those of iBTKs, dBTKs require more training data to excel its capability, otherwise it will suffer from the data sparseness problem. The reason that dBTK outperforms iBTK in the feature space of Root in *FBIS* corpus is that although it is a joint feature space, the Root node pairs can be constructed from a close set of grammar tags and to form a relatively low dimensional space.

  As a result, when applying to *FBIS* corpus, which only contains limited amount of training data, dBTKs will suffer more from the data sparseness problem, and therefore, a relatively low performance. When enlarging the amount of training corpus to the *HIT* corpus, the ability of dBTKs excels and the benefit from data increasing of dBTKs is more significant than iBTKs.
- We also find that the introduction of BTKs gains more improvement in HIT gold standard corpus than in FBIS corpus. Other than the factor of the amount of training data, this is also because the plain features in Table 3 are not as effective as those in Table 4, since they are trained on FBIS corpus which facilitates Table 4 more with respect to the domains. On the other hand, the grammatical tags and syntactic tree structures are more accurate in HIT corpus, which facilitates the performance of BTKs in Table 3.
- On the comparison across the different feature spaces of BTKs, we find that STT, RdSTT and TgSTT are rather selective, since **Recall**s of those feature spaces are relatively low, exp. for HIT corpus. However, the Root sub-structure obtains a satisfactory **Recall** for both corpora. That's why we attempt to construct a more complex composite kernel in adoption of the kernel of dBTK-Root as below.
- To gain an extra performance boosting, we further construct a composite kernel which includes the best iBTK and the best dBTK for each corpus along with the polynomial kernel for plain features. In the HIT corpus, we use dBTK in the Root space and iBTK in the RgSST space; while for FBIS corpus, we use dBTK in the Root space and iBTK in the RdSST space. The experimental results suggest that by combining iBTK and dBTK together, we can achieve more improvement.

7 Experiments on Machine Translation

In addition to the intrinsic alignment evaluation, we further conduct the extrinsic MT evaluation. We explore the effectiveness of sub-tree alignment for both phrase based and linguistically motivated syntax based SMT systems.

7.1 Experimental configuration

In the experiments, we train the translation model on FBIS corpus (7.2M (Chinese) + 9.2M (English) words in 240,000 sentence pairs) and train a 4-gram language model on the Xinhua portion of the English Gigaword corpus (181M words) using the SRILM Toolkits (Stolcke, 2002). We use these sentences with less than 50 characters from the NIST MT-2002 test set as the development set (to speed up tuning for syntax based system) and the NIST MT-2005 test set as our test set. We use the Stanford parser (Klein and Manning, 2003) to parse bilingual sentences on the training set and Chinese sentences on the development and test set. The evaluation metric is case-sensitive BLEU-4.

System	Model	BLEU
Moses	BP*	23.86
	DirC	23.98
	EWoS	24.48
Syntax STSG	STSG	24.71
	DirC	25.16
	EWoS	25.38
Syntax STSSG	STSSG	25.92
	DirC	25.95
	EWoS	26.45

Table 5. MT evaluation on various systems
*BP denotes bilingual phrases

For the phrase based system, we use Moses (Koehn et al., 2007) with its default settings. For the syntax based system, since sub-tree alignment can directly benefit Tree-2-Tree based systems, we apply the sub-tree alignment in a syntax system based on Synchronous Tree Substitution Grammar (STSG) (Zhang et al., 2007). The STSG based decoder uses a pair of *elementary tree*[3] as a basic translation unit. Recent research on tree based systems shows that relaxing the restriction from tree structure to tree sequence structure (Synchronous Tree Sequence Substitution Grammar: STSSG) significantly improves the translation performance (Zhang et al., 2008). We implement the STSG/STSSG based model in the Pisces decoder with the identical features and settings in Sun et al. (2009). In the Pisces decoder, the STSSG based decoder translates each span iteratively in a bottom up manner which guarantees that when translating a source span, any of its sub-spans is already translated. The STSG based decoding can be easily performed with the STSSG decoder by restricting the translation rule set to be elementary tree pairs only.

As for the alignment setting, we use the word alignment trained on the entire FBIS (240k) corpus by GIZA++ with heuristic grow-diag-final for both Moses and the syntax system. For sub-tree-alignment, we use the above word alignment to learn lexical/word alignment feature, and train with the FBIS training corpus (200) using the composite kernel of Plain+dBTK-Root+iBTK-RdSTT.

7.2 Experimental results

Compared with the adoption of word alignment, translational equivalences generated from structural alignment tend to be more grammatically aware and syntactically meaningful. However, utilizing syntactic translational equivalences alone for machine translation loses the capability of modeling non-syntactic phrases (Koehn et al., 2003). Consequently, instead of using phrases constraint by sub-tree alignment alone, we attempt to combine word alignment and sub-tree alignment and deploy the capability of both with two methods.

- *Directly Concatenate* (DirC) is operated by directly concatenating the rule set genereted from sub-tree alignment and the original rule set generated from word alignment (Tinsley et al., 2009). As shown in Table 5, we gain minor improvement in the Bleu score for all configurations.
- Alternatively, we proposed a new approach to generate the rule set from the scratch. We constrain the bilingual phrases to be consistent with *Either Word alignment or Sub-tree alignment* (EWoS) instead of being originally consistent with the word alignment only. The method helps tailoring the rule set decently without redundant counts for syntactic rules. The performance is further improved compared to DirC in all systems.

The findings suggest that with the modeling of non-syntactic phrases maintained, more emphasis on syntactic phrases can benefit both the phrase and syntax based SMT systems.

8 Conclusion

In this paper, we explore syntactic structure features by means of Bilingual Tree Kernels and apply them to bilingual sub-tree alignment along with various lexical and plain structural features. We use both gold standard tree bank and the automatically parsed corpus for the sub-tree alignment evaluation. Experimental results show that our model significantly outperforms the baseline method and the proposed Bilingual Tree Kernels are very effective in capturing the cross-lingual structural similarity. Further experiment shows that the obtained sub-tree alignment benefits both phrase and syntax based MT systems by delivering more weight on syntactic phrases.

Acknowledgments
We thank MITLAB[4] in Harbin Institute of Technology for licensing us their sub-tree alignment corpus for our research.

[3] An elementary tree is a fragment whose leaf nodes can be either non-terminal symbols or terminal symbols.

[4] http://mitlab.hit.edu.cn/ .

References

David Burkett and Dan Klein. 2008. *Two languages are better than one* (*for syntactic parsing*). In Proceedings of EMNLP-08. 877-886.

Nello Cristianini and John Shawe-Taylor. 2000. *An introduction to support vector machines and other kernelbased learning methods*. Cambridge: Cambridge University Press.

Michael Collins and Nigel Duffy. 2001. *Convolution Kernels for Natural Language*. In Proceedings of NIPS-01.

Declan Groves, Mary Hearne and Andy Way. 2004. *Robust sub-sentential alignment of phrase-structure trees*. In Proceedings of COLING-04, pages 1072-1078.

Kenji Imamura. 2001. *Hierarchical Phrase Alignment Harmonized with Parsing*. In Proceedings of NLPRS-01, Tokyo. 377-384.

Thorsten Joachims. 1999. *Making large-scale SVM learning practical*. In B. SchÄolkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning, MIT press.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. In Proceedings of ACL-03. 423-430.

Philipp Koehn, Franz Josef Och and Daniel Marcu. 2003. *Statistical phrase-based translation*. In Proceedings of HLT-NAACL-03. 48-54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation*. In Proceedings of ACL-07. 177-180.

Franz Josef Och and Hermann Ney. 2003. *A systematic comparison of various statistical alignment models*. Computational Linguistics, 29(1):19-51, March.

Alessandro Moschitti. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. In Proceedings of ACL-04.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. In Proceedings of ICSLP-02. 901-904.

Jun Sun, Min Zhang and Chew Lim Tan. 2009. *A non-contiguous Tree Sequence Alignment-based Model for Statistical Machine Translation*. In Proceedings of ACL-IJCNLP-09. 914-922.

John Tinsley, Ventsislav Zhechev, Mary Hearne and Andy Way. 2007. *Robust language pair-independent sub-tree alignment*. In Proceedings of MT Summit XI -07.

John Tinsley, Mary Hearne and Andy Way. 2009. *Parallel treebanks in phrase-based statistical machine translation*. In Proceedings of CICLING-09.

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features*. In Proceedings of ACL-COLING-06. 825-832.

Min Zhang, Hongfei Jiang, AiTi Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A tree-to-tree alignment-based model for statistical machine translation*. In Proceedings of MT Summit XI -07. 535-542.

Min Zhang, Hongfei Jiang, AiTi Aw, Haizhou Li, Chew Lim Tan and Sheng Li. 2008. *A tree sequence alignment-based tree-to-tree translation model*. In Proceedings of ACL-08. 559-567.

Discriminative Pruning for Discriminative ITG Alignment

Shujie Liu[†], **Chi-Ho Li**[‡] and **Ming Zhou**[‡]
[†]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
shujieliu@mtlab.hit.edu.cn
[‡]Microsoft Research Asia, Beijing, China
{chl, mingzhou}@microsoft.com

Abstract

While Inversion Transduction Grammar (ITG) has regained more and more attention in recent years, it still suffers from the major obstacle of speed. We propose a discriminative ITG pruning framework using Minimum Error Rate Training and various features from previous work on ITG alignment. Experiment results show that it is superior to all existing heuristics in ITG pruning. On top of the pruning framework, we also propose a discriminative ITG alignment model using hierarchical phrase pairs, which improves both F-score and Bleu score over the baseline alignment system of GIZA++.

1 Introduction

Inversion transduction grammar (ITG) (Wu, 1997) is an adaptation of SCFG to bilingual parsing. It does synchronous parsing of two languages with phrasal and word-level alignment as by-product. For this reason ITG has gained more and more attention recently in the word alignment community (Zhang and Gildea, 2005; Cherry and Lin, 2006; Haghighi *et al.*, 2009).

A major obstacle in ITG alignment is speed. The original (unsupervised) ITG algorithm has complexity of $O(n^6)$. When extended to supervised/discriminative framework, ITG runs even more slowly. Therefore all attempts to ITG alignment come with some pruning method. For example, Haghighi *et al.* (2009) do pruning based on the probabilities of links from a simpler alignment model (viz. HMM); Zhang and Gildea (2005) propose Tic-tac-toe pruning, which is based on the Model 1 probabilities of word pairs inside and outside a pair of spans.

As all the principles behind these techniques have certain contribution in making good pruning decision, it is tempting to incorporate all these features in ITG pruning. In this paper, we propose a novel discriminative pruning framework for discriminative ITG. The pruning model uses no more training data than the discriminative ITG parser itself, and it uses a log-linear model to integrate all features that help identify the correct span pair (like Model 1 probability and HMM posterior). On top of the discriminative pruning method, we also propose a discriminative ITG alignment system using hierarchical phrase pairs.

In the following, some basic details on the ITG formalism and ITG parsing are first reviewed (Sections 2 and 3), followed by the definition of pruning in ITG (Section 4). The "Discriminative Pruning for Discriminative ITG" model (DPDI) and our discriminative ITG (DITG) parsers will be elaborated in Sections 5 and 6 respectively. The merits of DPDI and DITG are illustrated with the experiments described in Section 7.

2 Basics of ITG

The simplest formulation of ITG contains three types of rules: terminal unary rules $X \rightarrow e/f$, where e and f represent words (possibly a null word, ε) in the English and foreign language respectively, and the binary rules $X \rightarrow [X, X]$ and $X \rightarrow \langle X, X \rangle$, which refer to that the component English and foreign phrases are combined in the same and inverted order respectively.

From the viewpoint of word alignment, the terminal unary rules provide the links of word pairs, whereas the binary rules represent the reordering factor. One of the merits of ITG is that it is less biased towards short-distance reordering.

Such a formulation has two drawbacks. First of all, it imposes a 1-to-1 constraint in word alignment. That is, a word is not allowed to align to more than one word. This is a strong limitation as no idiom or multi-word expression is allowed to align to a single word on the other side. In fact there have been various attempts in relaxing the 1-to-1 constraint. Both ITG alignment

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 316–324,
Uppsala, Sweden, 11-16 July 2010.

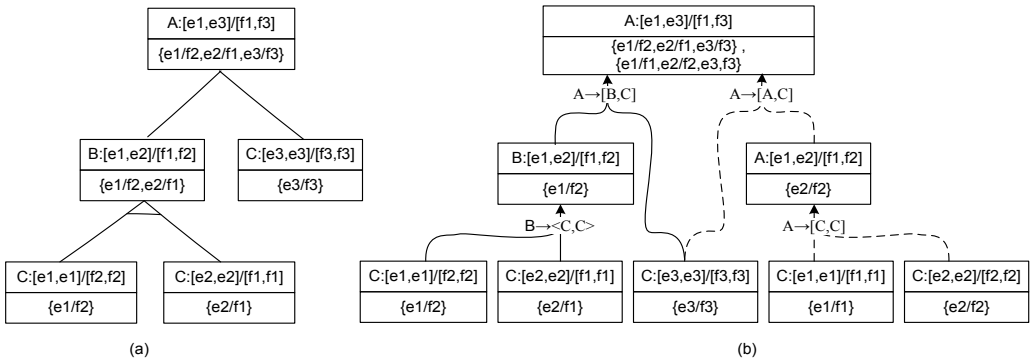


Figure 1: Example ITG parses in graph (a) and hypergraph (b).

approaches with and without this constraint will be elaborated in Section 6.

Secondly, the simple ITG leads to redundancy if word alignment is the sole purpose of applying ITG. For instance, there are two parses for three consecutive word pairs, viz. $[a/a'\,[b/b'\,c/c']\,]$ and $[[a/a'\,b/b']\,c/c']$. The problem of redundancy is fixed by adopting ITG normal form. In fact, normal form is the very first key to speeding up ITG. The ITG normal form grammar as used in this paper is described in Appendix A.

3 Basics of ITG Parsing

Based on the rules in normal form, ITG word alignment is done in a similar way to chart parsing (Wu, 1997). The base step applies all relevant terminal unary rules to establish the links of word pairs. The word pairs are then combined into span pairs in all possible ways. Larger and larger span pairs are recursively built until the sentence pair is built.

Figure 1(a) shows one possible derivation for a toy example sentence pair with three words in each sentence. Each node (rectangle) represents a pair, marked with certain phrase category, of foreign span (F-span) and English span (E-span) (the upper half of the rectangle) and the associated alignment hypothesis (the lower half). Each graph like Figure 1(a) shows only one derivation and also only one alignment hypothesis.

The various derivations in ITG parsing can be compactly represented in hypergraph (Klein and Manning, 2001) like Figure 1(b). Each hypernode (rectangle) comprises both a span pair (upper half) and the list of possible alignment hypotheses (lower half) for that span pair. The hyperedges show how larger span pairs are derived from smaller span pairs. Note that a hypernode may have more than one alignment hypothesis, since a hypernode may be derived through more than one hyperedge (e.g. the topmost hypernode in Figure 1(b)). Due to the use of normal form, the hypotheses of a span pair are different from each other.

4 Pruning in ITG Parsing

The ITG parsing framework has three levels of pruning:

1) To discard some unpromising span pairs;
2) To discard some unpromising F-spans and/or E-spans;
3) To discard some unpromising alignment hypotheses for a particular span pair.

The second type of pruning (used in Zhang *et. al.* (2008)) is very radical as it implies discarding too many span pairs. It is empirically found to be highly harmful to alignment performance and therefore not adopted in this paper.

The third type of pruning is equivalent to minimizing the beam size of alignment hypotheses in each hypernode. It is found to be well handled by the K-Best parsing method in Huang and Chiang (2005). That is, during the bottom-up construction of the span pair repertoire, each span pair keeps only the best alignment hypothesis. Once the complete parse tree is built, the k-best list of the topmost span is obtained by minimally expanding the list of alignment hypotheses of minimal number of span pairs.

The first type of pruning is equivalent to minimizing the number of hypernodes in a hypergraph. The task of ITG pruning is defined in this paper as the first type of pruning; i.e. the search for, given an F-span, the minimal number of E-spans which are the most likely counterpart of that F-span.[1] The pruning method should maintain a balance between efficiency (run as quickly as possible) and performance (keep as many correct span pairs as possible).

[1] Alternatively it can be defined as the search of the minimal number of E-spans per F-span. That is simply an arbitrary decision on how the data are organized in the ITG parser.

A naïve approach is that the required pruning method outputs a score given a span pair. This score is used to rank all E-spans for a particular F-span, and the score of the correct E-span should be in general higher than most of the incorrect ones.

5 The DPDI Framework

DPDI, the discriminative pruning model proposed in this paper, assigns score to a span pair $(\bar{f}, \bar{e})$ as probability from a log-linear model:

$$P(\bar{e}|\bar{f}) = \frac{exp(\sum_i \lambda_i \Psi_i(\bar{f}, \bar{e}))}{\sum_{\bar{e}' \in E} exp(\sum_i \lambda_i \Psi_i(\bar{f}, \bar{e}'))} \quad (1)$$

where each $\Psi_i(\bar{f}, \bar{e})$ is some feature about the span pair, and each λ is the weight of the corresponding feature. There are three major questions to this model:

1) How to acquire training samples? (Section 5.1)
2) How to train the parameters λ ? (Section 5.2)
3) What are the features? (Section 5.3)

5.1 Training Samples

Discriminative approaches to word alignment use manually annotated alignment for sentence pairs. Discriminative pruning, however, handles not only a sentence pair but every possible span pair. The required training samples consist of various F-spans and their corresponding E-spans.

Rather than recruiting annotators for marking span pairs, we modify the parsing algorithm in Section 3 so as to produce span pair annotation out of sentence-level annotation. In the base step, only the word pairs listed in sentence-level annotation are inserted in the hypergraph, and the recursive steps are just the same as usual.

If the sentence-level annotation satisfies the alignment constraints of ITG, then each F-span will have only one E-span in the parse tree. However, in reality there are often the cases where a foreign word aligns to more than one English word. In such cases the F-span covering that foreign word has more than one corresponding E-spans. Consider the example in Figure 2, where the golden links in the alignment annotation are $e1/f1$, $e2/f1$, and $e3/f2$; i.e. the foreign word $f1$ aligns to both the English words $e1$ and $e2$. Therefore the F-span $[f1, f1]$ aligns to the E-span $[e1, e1]$ in one hypernode and to the E-span $[e2, e2]$ in another hypernode. When such situation happens, we calculate the product of the inside and outside probability of each alignment hypothesis of the span pair, based on the probabilities of the links from some simpler alignment model[2]. The E-span with the most probable hypothesis is selected as the alignment of the F-span.

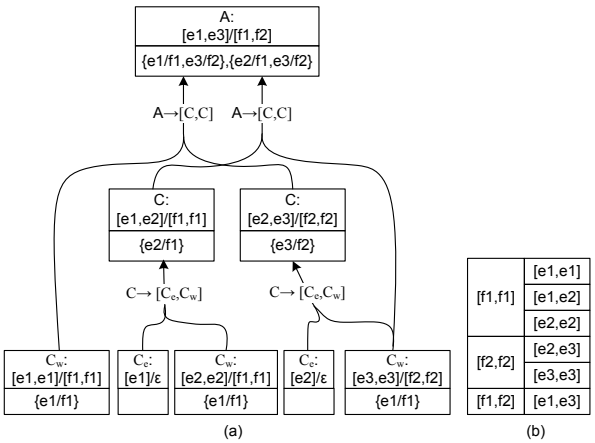


Figure 2: Training sample collection.

Table (b) lists, for the hypergraph in (a), the candidate E-spans for each F-span.

It should be noted that this automatic span pair annotation may violate some of the links in the original sentence-level alignment annotation. We have already seen how the 1-to-1 constraint in ITG leads to the violation. Another situation is the 'inside-out' alignment pattern (c.f. Figure 3). The ITG reordering constraint cannot be satisfied unless one of the links in this pattern is removed.

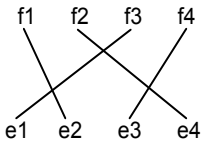


Figure 3: An example of inside-out alignment

The training samples thus obtained are positive training samples. If we apply some classifier for parameter training, then negative samples are also needed. Fortunately, our parameter training does not rely on any negative samples.

5.2 MERT for Pruning

Parameter training of DPDI is based on Minimum Error Rate Training (MERT) (Och, 2003), a widely used method in SMT. MERT for SMT estimates model parameters with the objective of minimizing certain measure of translation errors (or maximizing certain performance measure of translation quality) for a development corpus. Given an SMT system which produces, with

[2] The formulae of the inside and outside probability of a span pair will be elaborated in Section 5.3. The simpler alignment model we used is HMM.

model parameters λ_1^M, the K-best candidate translations $\hat{e}(f_s; \lambda_1^M)$ for a source sentence f_s, and an error measure $E(r_s, e_{s,k})$ of a particular candidate $e_{s,k}$ with respect to the reference translation r_s, the optimal parameter values will be:

$$\hat{\lambda}_1^M = \underset{\lambda_1^M}{argmin}\left\{\sum_{s=1}^{S} E\left(r_s, \hat{e}(f_s; \lambda_1^M)\right)\right\}$$

$$= \underset{\lambda_1^M}{argmin}\left\{\sum_{s=1}^{S}\sum_{k=1}^{K} E\left(r_s, e_{s,k}\right)\delta(\hat{e}(f_s; \lambda_1^M), e_{s,k})\right\}$$

DPDI applies the same equation for parameter tuning, with different interpretation of the components in the equation. Instead of a development corpus with reference translations, we have a collection of training samples, each of which is a pair of F-span (f_s) and its corresponding E-span (r_s). These samples are acquired from some manually aligned dataset by the method elaborated in Section 5.1. The ITG parser outputs for each f_s a K-best list of E-spans $\hat{e}(f_s; \lambda_1^M)$ based on the current parameter values λ_1^M.

The error function is based on the presence and the rank of the correct E-span in the K-best list:

$$E\left(r_s, \hat{e}(f_s; \lambda_1^M)\right) = \begin{cases} -rank(r_s) \; if \; r_s \in \hat{e}(f_s; \lambda_1^M) \\ penalty \quad otherwise \end{cases} \tag{2}$$

where $rank(r_s)$ is the (0-based) rank of the correct E-span r_s in the K-best list $\hat{e}(f_s; \lambda_1^M)$. If r_s is not in the K-best list at all, then the error is defined to be $penalty$, which is set as *-100000* in our experiments. The rationale underlying this error function is to keep as many correct E-spans as possible in the K-best lists of E-spans, and push the correct E-spans upward as much as possible in the K-best lists.

This new error measure leads to a change in details of the training algorithm. In MERT for SMT, the interval boundaries at which the performance or error measure changes are defined by the *upper envelope* (illustrated by the dash line in Figure 4(a)), since the performance/error measure depends on the best candidate translation. In MERT for DPDI, however, the error measure depends on the correct E-span rather than the E-span leading to the highest system score. Thus the interval boundaries are the intersections between the correct E-span and all other candidate E-spans (as shown in Figure 4(b)). The rank of the correct E-span in each interval can then be figured out as shown in Figure 4(c). Finally, the error measure in each interval can be calculated by Equation (2) (as shown in Figure 4(d)). All other steps in MERT for DPDI are the same as that for SMT.

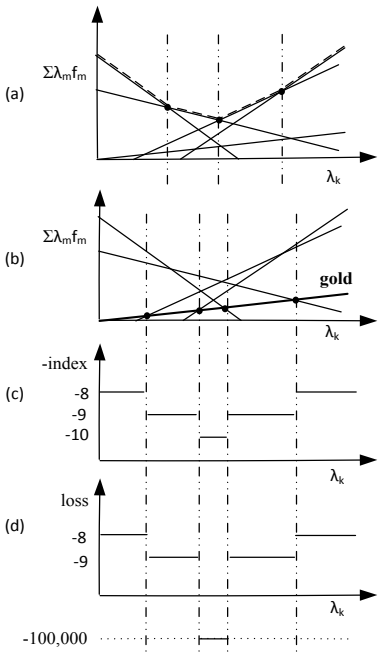


Figure 4: MERT for DPDI

Part (a) shows how intervals are defined for SMT and part (b) for DPDI. Part (c) obtains the rank of correct E-spans in each interval and part (d) the error measure. Note that the beam size (max number of E-spans) for each F-span is 10.

5.3 Features

The features used in DPDI are divided into three categories:

1) Model 1-based probabilities. Zhang and Gildea (2005) show that Model 1 (Brown *et al.*, 1993; Och and Ney., 2000) probabilities of the word pairs inside and outside a span pair ($[e_{i1}, e_{i2}]/[f_{j1}, f_{j2}]$) are useful. Hence these two features:
   a) Inside probability (i.e. probability of word pairs within the span pair):

$$p_{inc}\left(e_{i1,i2}|f_{j1,j2}\right) = \prod_{i\in(i1,i2)}\sum_{j\in(j1,j2)}\frac{1}{(j2-j1)}p_{M1}(e_i|f_j)$$

   b) Outside probability (i.e. probability of the word pairs outside the span pair):

$$p_{out}\left(e_{i1,i2}|f_{j1,j2}\right) = \prod_{i\notin(i1,i2)}\sum_{j\notin(j1,j2)}\frac{1}{(J-j2+j1)}p_{M1}(e_i|f_j)$$

   where J is the length of the foreign sentence.
2) Heuristics. There are four features in this category. The features are explained with the

example of Figure 5, in which the span pair in interest is $[e2, e3]/[f1, f2]$. The four links are produced by some simpler alignment model like HMM. The word pair $e2/f1$ is the only link in the span pair. The links $e4/f2$ and $e3/f3$ are inconsistent with the span pair.[3]

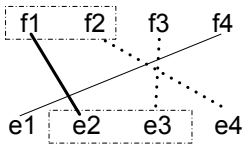


Figure 5: Example for heuristic features

a) Link ratio: $\frac{2\times\#links}{flen+elen}$
where $\#links$ is the number of links in the span pair, and $flen$ and $elen$ are the length of the foreign and English spans respectively. The feature value of the example span pair is (2*1)/(2+2)=0.5.

b) inconsistent link ratio: $\frac{2\times\#links_{incon}}{flen+elen}$
where $\#links_{incon}$ is the number of links which are inconsistent with the phrase pair according to some simpler alignment model (e.g. HMM). The feature value of the example is (2*2)/(2+2) =1.0.

c) Length ratio: $\left|\frac{flen}{elen} - ratio_{avg}\right|$
where $ratio_{avg}$ is defined as the average ratio of foreign sentence length to English sentence length, and it is estimated to be around 1.15 in our training dataset. The rationale underlying this feature is that the ratio of span length should not be too deviated from the average ratio of sentence length. The feature value for the example is |2/2-1.15|=0.15.

d) Position Deviation: $\left|pos_{\bar{f}} - pos_{\bar{e}}\right|$
where $pos_{\bar{f}}$ refers to the position of the F-span in the entire foreign sentence, and it is defined as $\frac{1}{2J}\left(start_{\bar{f}} + end_{\bar{f}}\right)$, $start_{\bar{f}}$ /$end_{\bar{f}}$ being the position of the first/last word of the F-span in the foreign sentence. $pos_{\bar{e}}$ is defined similarly. The rationale behind this feature is the monotonic assumption, i.e. a phrase of the foreign sentence usually occupies roughly the same position of the equivalent English phrase. The feature value for the example is |(1+2)/(2*4)-(2+3)/(2*4)| =0.25.

3) HMM-based probabilities. Haghighi *et al.* (2009) show that posterior probabilities from the HMM alignment model is useful for pruning. Therefore, we design two new features by replacing the link count in link ratio and inconsistent link ratio with the sum of the link's posterior probability.

6 The DITG Models

The discriminative ITG alignment can be conceived as a two-staged process. In the first stage DPDI selects good span pairs. In the second stage good alignment hypotheses are assigned to the span pairs selected by DPDI. Two discriminative ITG (DITG) models are investigated. One is word-to-word DITG (henceforth W-DITG), which observes the 1-to-1 constraint on alignment. Another is DITG with hierarchical phrase pairs (henceforth HP-DITG), which relaxes the 1-to-1 constraint by adopting hierarchical phrase pairs in Chiang (2007).

Each model selects the best alignment hypotheses of each span pair, given a set of features. The contributions of these features are integrated through a log linear model (similar to Liu *et al.*, 2005; Moore, 2005) like Equation (1). The discriminative training of the feature weights is again MERT (Och, 2003). The MERT module for DITG takes alignment F-score of a sentence pair as the performance measure. Given an input sentence pair and the reference annotated alignment, MERT aims to maximize the F-score of DITG-produced alignment. Like SMT (and unlike DPDI), it is the upper envelope which defines the intervals where the performance measure changes.

6.1 Word-to-word DITG

The following features about alignment link are used in W-DITG:

1) Word pair translation probabilities trained from HMM model (Vogel, *et.al.*, 1996) and IBM model 4 (Brown *et.al.*, 1993; Och and Ney, 2000).
2) Conditional link probability (Moore, 2005).
3) Association score rank features (Moore *et al.*, 2006).
4) Distortion features: counts of inversion and concatenation.
5) Difference between the relative positions of the words. The relative position of a word in a sentence is defined as the posi-

[3] An inconsistent link connects a word within the phrase pair to some word outside the phrase pair. C.f. Deng *et al.* (2008)

tion of the word divided by sentence length.

6) Boolean features like whether a word in the word pair is a stop word.

6.2 DITG with Hierarchical Phrase Pairs

The 1-to-1 assumption in ITG is a serious limitation as in reality there are always segmentation or tokenization errors as well as idiomatic expressions. Wu (1997) proposes a bilingual segmentation grammar extending the terminal rules by including phrase pairs. Cherry and Lin (2007) incorporate phrase pairs in phrase-based SMT into ITG, and Haghighi *et al.* (2009) introduce Block ITG (BITG), which adds 1-to-many or many-to-1 terminal unary rules.

It is interesting to see if DPDI can benefit the parsing of a more realistic ITG. HP-DITG extends Cherry and Lin's approach by not only employing simple phrase pairs but also hierarchical phrase pairs (Chiang, 2007). The grammar is enriched with rules of the format: $X \rightarrow \bar{e}_i / \bar{f}_i$ where $\bar{e}_i$ and $\bar{f}_i$ refer to the English and foreign side of the *i*-th (simple/hierarchical) phrase pair respectively.

As example, if there is a simple phrase pair $X \rightarrow \langle North\ Korea, 北\ 朝鲜 \rangle$, then it is transformed into the ITG rule $C \rightarrow$ "*North Korea*"/"北 朝鲜". During parsing, each span pair does not only examine all possible combinations of sub-span pairs using binary rules, but also checks if the yield of that span pair is exactly the same as that phrase pair. If so, then the alignment links within the phrase pair (which are obtained in standard phrase pair extraction procedure) are taken as an alternative alignment hypothesis of that span pair.

For a hierarchical phrase pair like $X \rightarrow \langle X_1\ of\ X_2, X_2\ 的\ X_1 \rangle$, it is transformed into the ITG rule $C \rightarrow$ "X_1 *of* X_2"/"X_2 的 X_1" during parsing, each span pair checks if it contains the lexical anchors *"of"* and "的", and if the remaining words in its yield can form two sub-span pairs which fit the reordering constraint among X_1 and X_2. (Note that span pairs of any category in the ITG normal form grammar can substitute for X_1 or X_2.) If both conditions hold, then the span pair is assigned an alignment hypothesis which combines the alignment links among the lexical anchors ($like\ of$/的) and those links among the sub-span pairs.

HP-ITG acquires the rules from HMM-based word-aligned corpus using standard phrase pair extraction as stated in Chiang (2007). The rule probabilities and lexical weights in both English-to-foreign and foreign-to-English directions are estimated and taken as features, in addition to those features in W-DITG, in the discriminative model of alignment hypothesis selection.

7 Evaluation

DPDI is evaluated against the baselines of Tic-tac-toe (TTT) pruning (Zhang and Gildea, 2005) and Dynamic Program (DP) pruning (Haghighi *et al.*, 2009; DeNero *et al.*, 2009) with respect to Chinese-to-English alignment and translation. Based on DPDI, HP-DITG is evaluated against the alignment systems GIZA++ and BITG.

7.1 Evaluation Criteria

Four evaluation criteria are used in addition to the time spent on ITG parsing. We will first evaluate pruning regarding the pruning decisions themselves. That is, the first evaluation metric, pruning error rate (henceforth PER), measures how many correct E-spans are discarded. The major drawback of PER is that not all decisions in pruning would impact on alignment quality, since certain F-spans are of little use to the entire ITG parse tree.

An alternative criterion is the upper bound on alignment F-score, which essentially measures how many links in annotated alignment can be kept in ITG parse. The calculation of F-score upper bound is done in a bottom-up way like ITG parsing. All leaf hypernodes which contain a correct link are assigned a score (known as hit) of 1. The hit of a non-leaf hypernode is based on the sum of hits of its daughter hypernodes. The maximal sum among all hyperedges of a hypernode is assigned to that hypernode. Formally,

$$hit\big(X[\bar{f}, \bar{e}]\big) = \max_{Y,Z,\bar{f}_1,\bar{e}_1,\bar{f}_2,\bar{e}_2} \big(hit\big(Y[\bar{f}_1, \bar{e}_1]\big) + hit[\bar{f}_2, \bar{e}_2]\big)$$

$$hit(C_w[u, v]) = \begin{cases} 1 & if\ \langle u, v \rangle \in R \\ 0 & otherwise \end{cases}$$

$$hit(C_e) = 0; hit\big(C_f\big) = 0$$

where X, Y, Z are variables for the categories in ITG grammar, and R comprises the golden links in annotated alignment. C_w, C_e, C_f are defined in Appendix A.

Figure 6 illustrates the calculation of the hit score for the example in Section 5.1/Figure 2. The upper bound of recall is the hit score divided by the total number of golden links. The upper

ID	pruning	beam size	pruning/total time cost	PER	F-UB	F-score
1	DPDI	10	72''/3'03''	**4.9%**	**88.5%**	**82.5%**
2	TTT	10	**58''/2'38''**	8.6%	87.5%	81.1%
3	TTT	20	53''/6'55''	5.2%	**88.6%**	82.4%
4	DP	--	11''/6'01''	12.1%	86.1%	80.5%

Table 1: Evaluation of DPDI against TTT (Tic-tac-toe) and DP (Dynamic Program) for W-DITG

ID	pruning	beam size	pruning/total time cost	PER	F-UB	F-score
1	DPDI	10	72''/5'18''	**4.9%**	**93.9%**	**87.0%**
2	TTT	10	**58''/4'51''**	8.6%	93.0%	84.8%
3	TTT	20	53''/12'5''	5.2%	**94.0%**	86.5%
4	DP	--	11''/15'39''	12.1%	91.4%	83.6%

Table 2: Evaluation of DPDI against TTT (Tic-tac-toe) and DP (Dynamic Program) for HP-DITG.

bound of precision, which should be defined as the hit score divided by the number of links produced by the system, is almost always 1.0 in practice. The upper bound of alignment F-score can thus be calculated as well.

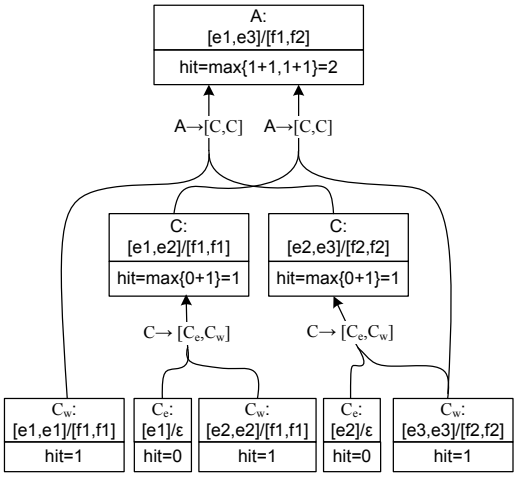


Figure 6: Recall Upper Bound Calculation

Finally, we also do end-to-end evaluation using both F-score in alignment and Bleu score in translation. We use our implementation of hierarchical phrase-based SMT (Chiang, 2007), with standard features, for the SMT experiments.

7.2 Experiment Data

Both discriminative pruning and alignment need training data and test data. We use the manually aligned Chinese-English dataset as used in Haghighi *et al.* (2009). The 491 sentence pairs in this dataset are adapted to our own Chinese word segmentation standard. 250 sentence pairs are used as training data and the other 241 are test data. The corresponding numbers of F-spans in training and test data are 4590 and 3951 respectively.

In SMT experiments, the bilingual training dataset is the NIST training set excluding the Hong Kong Law and Hong Kong Hansard, and our 5-gram language model is trained from the Xinhua section of the Gigaword corpus. The NIST'03 test set is used as our development corpus and the NIST'05 and NIST'08 test sets are our test sets.

7.3 Small-scale Evaluation

The first set of experiments evaluates the performance of the three pruning methods using the small 241-sentence set. Each pruning method is plugged in both W-DITG and HP-DITG. IBM Model 1 and HMM alignment model are re-implemented as they are required by the three ITG pruning methods.

The results for W-DITG are listed in Table 1. Tests 1 and 2 show that with the same beam size (i.e. number of E-spans per F-span), although DPDI spends a bit more time (due to the more complicated model), DPDI makes far less incorrect pruning decisions than the TTT. In terms of F-score upper bound, DPDI is 1 percent higher. DPDI achieves even larger improvement in actual F-score.

To enable TTT achieving similar F-score or F-score upper bound, the beam size has to be doubled and the time cost is more than twice the original (c.f. Tests 1 and 3 in Table 1) .

The DP pruning in Haghighi *et.al.* (2009) performs much poorer than the other two pruning methods. In fact, we fail to enable DP achieve the same F-score upper bound as the other two methods before DP leads to intolerable memory consumption. This may be due to the use of different HMM model implementations between our work and Haghighi *et.al.* (2009).

Table 2 lists the results for HP-DITG. Roughly the same observation as in W-DITG can be made. In addition to the superiority of DPDI, it can also be noted that HP-DITG achieves much higher F-score and F-score upper bound. This shows that

hierarchical phrase is a powerful tool in rectifying the 1-to-1 constraint in ITG.

Note also that while TTT in Test 3 gets roughly the same F-score upper bound as DPDI in Test 1, the corresponding F-score is slightly worse. A possible explanation is that better pruning not only speeds up the parsing/alignment process but also guides the search process to focus on the most promising region of the search space.

7.4 Large-scale End-to-End Experiment

ID	Pruning	beam size	time cost	Bleu-05	Bleu-08
1	DPDI	10	1092h	**38.57**	**28.31**
2	TTT	10	**972h**	37.96	27.37
3	TTT	20	2376h	38.13	27.58
4	DP	--	2068h	37.43	27.12

Table 3: Evaluation of DPDI against TTT and DP for HP-DITG

ID	WA-Model	F-Score	Bleu-05	Bleu-08
1	HMM	80.1%	36.91	26.86
2	Giza++	84.2%	37.70	27.33
3	BITG	85.9%	37.92	27.85
4	HP-DITG	**87.0%**	**38.57**	**28.31**

Table 4: Evaluation of DPDI against HMM, Giza++ and BITG

Table 3 lists the word alignment time cost and SMT performance of different pruning methods. HP-DITG using DPDI achieves the best Bleu score with acceptable time cost. Table 4 compares HP-DITG to HMM (Vogel, et al., 1996), GIZA++ (Och and Ney, 2000) and BITG (Haghighi *et al.*, 2009). It shows that HP-DITG (with DPDI) is better than the three baselines both in alignment F-score and Bleu score. Note that the Bleu score differences between HP-DITG and the three baselines are statistically significant (Koehn, 2004).

An explanation of the better performance by HP-DITG is the better phrase pair extraction due to DPDI. On the one hand, a good phrase pair often fails to be extracted due to a link inconsistent with the pair. On the other hand, ITG pruning can be considered as phrase pair selection, and good ITG pruning like DPDI guides the subsequent ITG alignment process so that less links inconsistent to good phrase pairs are produced. This also explains (in Tables 2 and 3) why DPDI with beam size 10 leads to higher Bleu than TTT with beam size 20, even though both pruning methods lead to roughly the same alignment F-score.

8 Conclusion and Future Work

This paper reviews word alignment through ITG parsing, and clarifies the problem of ITG pruning. A discriminative pruning model and two discriminative ITG alignments systems are proposed. The pruning model is shown to be superior to all existing ITG pruning methods, and the HP-DITG alignment system is shown to improve state-of-the-art alignment and translation quality.

The current DPDI model employs a very limited set of features. Many features are related only to probabilities of word pairs. As the success of HP-DITG illustrates the merit of hierarchical phrase pair, in future we should investigate more features on the relationship between span pair and hierarchical phrase pair.

Appendix A. The Normal Form Grammar

Table 5 lists the ITG rules in normal form as used in this paper, which extend the normal form in Wu (1997) so as to handle the case of alignment to null.

1	$S \rightarrow A \mid B \mid C$
2	$A \rightarrow [A\ B] \mid [A\ C] \mid [B\ B] \mid [BC] \mid [C\ B] \mid [C\ C]$
3	$B \rightarrow \langle A\ A\rangle \mid \langle A\ C\rangle \mid \langle B\ A\rangle \mid \langle B\ C\rangle$
	$B \rightarrow \langle C\ A\rangle \mid \langle C\ C\rangle$
4	$C \rightarrow C_w \mid C_{fw} \mid C_{ew}$
5	$C \rightarrow [C_{ew}\ C_{fw}]$
6	$C_w \rightarrow u/v$
7	$C_e \rightarrow \varepsilon/v; C_f \rightarrow u/\varepsilon$
8	$C_{em} \rightarrow C_e \mid [C_{em}\ C_e]; C_{fm} \rightarrow C_f \mid [C_{fm}\ C_f]$
9	$C_{ew} \rightarrow [C_{em}\ C_w]; C_{fw} \rightarrow [C_{fm}\ C_w]$

Table 5: ITG Rules in Normal Form

In these rules, S is the Start symbol; A is the category for concatenating combination whereas B for inverted combination. Rules (2) and (3) are inherited from Wu (1997). Rules (4) divide the terminal category C into subcategories. Rule schema (6) subsumes all terminal unary rules for some English word u and foreign word v, and rule schemas (7) are unary rules for alignment to null. Rules (8) ensure all words linked to null are combined in left branching manner, while rules (9) ensure those words linked to null combine with some following, rather than preceding, word pair. (Note: Accordingly, all sentences must be ended by a special token $\langle end\rangle$, otherwise the last word(s) of a sentence cannot be linked to null.) If there are both English and foreign words linked to null, rule (5) ensures that those English

words linked to null precede those foreign words linked to null.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Peitra, Robert L. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics,* 19(2):263-311.

Colin Cherry and Dekang Lin. 2006. *Soft Syntactic Constraints for Word Alignment through Discriminative Training.* In *Proceedings of* ACL-COLING.

Colin Cherry and Dekang Lin. 2007. *Inversion Transduction Grammar for Joint Phrasal Translation Modeling.* In *Proceedings of* SSST, NAACL-HLT, Pages:17-24.

David Chiang. 2007. *Hierarchical Phrase-based Translation. Computational Linguistics*, 33(2).

John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. 2009. *Efficient Parsing for Transducer Grammars.* In *Proceedings of* NAACL, Pages:227-235.

Alexander Fraser and Daniel Marcu. 2006. *Semi-Supervised Training for StatisticalWord Alignment.* In *Proceedings of* ACL, Pages:769-776.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. *Better Word Alignments with Supervised ITG Models.* In *Proceedings of* ACL, Pages: 923-931.

Liang Huang and David Chiang. 2005. *Better k-best Parsing.* In *Proceedings of* IWPT 2005, Pages:173-180.

Franz Josef Och and Hermann Ney. 2000. *Improved statistical alignment models.* In *Proceedings of* ACL. Pages: 440-447

Franz Josef Och. 2003. *Minimum error rate training in statistical machine translation.* In *Proceedings of* ACL, Pages:160-167.

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs.* In *Proceedings of* IWPT, Pages:17-19

Philipp Koehn. 2004. *Statistical Significance Tests for Machine Translation Evaluation.* In *Proceedings of* EMNLP, Pages: 388-395.

Yang Liu, Qun Liu and Shouxun Lin. 2005. *Log-linear models for word alignment.* In *Proceedings of* ACL, Pages: 81-88.

Robert Moore. 2005. *A Discriminative Framework for Bilingual Word Alignment.* In *Proceedings of* EMNLP 2005, Pages: 81-88.

Robert Moore, Wen-tau Yih, and Andreas Bode. 2006. *Improved Discriminative Bilingual Word Alignment.* In *Proceedings of* ACL, Pages: 513-520.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. *HMM-based word alignment in statistical translation.* In *Proceedings of* COLING, Pages: 836-841.

Stephan Vogel. 2005. *PESA: Phrase Pair Extraction as Sentence Splitting.* In *Proceedings of* MT Summit.

Dekai Wu. 1997. *Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. Computational Linguistics*, 23(3).

Hao Zhang and Daniel Gildea. 2005. *Stochastic Lexicalized Inversion Transduction Grammar for Alignment.* In *Proceedings of* ACL.

Hao Zhang, Chris Quirk, Robert Moore, and Daniel Gildea. 2008. *Bayesian learning of non-compositional phrases with synchronous parsing.* In *Proceedings of* ACL, Pages: 314-323.

Fine-grained Tree-to-String Translation Rule Extraction

Xianchao Wu[†] **Takuya Matsuzaki**[†] **Jun'ichi Tsujii**[†‡*]

[†]Department of Computer Science, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
[‡]School of Computer Science, University of Manchester
[*]National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre, 131 Princess Street, Manchester M1 7DN, UK

{wxc, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

Tree-to-string translation rules are widely used in linguistically syntax-based statistical machine translation systems. In this paper, we propose to use *deep syntactic information* for obtaining fine-grained translation rules. A head-driven phrase structure grammar (HPSG) parser is used to obtain the deep syntactic information, which includes a fine-grained description of the syntactic property and a semantic representation of a sentence. We extract fine-grained rules from aligned HPSG tree/forest-string pairs and use them in our tree-to-string and string-to-tree systems. Extensive experiments on large-scale bidirectional Japanese-English translations testified the effectiveness of our approach.

1 Introduction

Tree-to-string translation rules are generic and applicable to numerous *linguistically syntax-based* Statistical Machine Translation (SMT) systems, such as string-to-tree translation (Galley et al., 2004; Galley et al., 2006; Chiang et al., 2009), tree-to-string translation (Liu et al., 2006; Huang et al., 2006), and forest-to-string translation (Mi et al., 2008; Mi and Huang, 2008). The algorithms proposed by Galley et al. (2004; 2006) are frequently used for extracting minimal and composed rules from aligned 1-best tree-string pairs. Dealing with the parse error problem and rule sparseness problem, Mi and Huang (2008) replaced the 1-best parse tree with a packed forest which compactly encodes exponentially many parses for tree-to-string rule extraction.

However, current tree-to-string rules only make use of Probabilistic Context-Free Grammar tree fragments, in which part-of-speech (POS) or phrasal tags are used as the tree node labels. As will be testified by our experiments, we argue that the simple POS/phrasal tags are too coarse to reflect the accurate translation probabilities of the translation rules.

	koroshita (active)	*korosareta* (passive)
VBN(*killed*)	**6** (6/10,6/6)	4 (4/10,4/4)
VBN(*killed*:active)	**5** (5/6,5/6)	1 (1/6,1/4)
VBN(*killed*:passive)	1 (1/4,1/6)	**3** (3/4,3/4)

Table 1: Bidirectional translation probabilities of rules, denoted in the brackets, change when *voice* is attached to "*killed*".

For example, as shown in Table 1, suppose a simple tree fragment "VBN(*killed*)" appears 6 times with "*koroshita*", which is a Japanese translation of an *active* form of "*killed*", and 4 times with "*korosareta*", which is a Japanese translation of a *passive* form of "*killed*". Then, without larger tree fragments, we will more frequently translate "VBN(*killed*)" into "*koroshita*" (with a probability of 0.6). But, "VBN(*killed*)" is indeed separable into two fine-grained tree fragments of "VBN(*killed*:active)" and "VBN(*killed*:passive)"[1]. Consequently, "VBN(*killed*:active)" appears 5 times with "*koroshita*" and 1 time with "*korosareta*"; and "VBN(*killed*:passive)" appears 1 time with "*koroshita*" and 3 times with "*korosareta*". Now, by attaching the *voice* information to "*killed*", we are gaining a rule set that is more appropriate to reflect the real translation situations.

This motivates our proposal of using deep syntactic information to obtain a fine-grained translation rule set. We name the information such as the voice of a verb in a tree fragment as deep syntactic information. We use a head-driven phrase structure grammar (HPSG) parser to obtain the

[1]For example, "*John has killed Mary.*" versus "*John was killed by Mary.*"

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 325–334,
Uppsala, Sweden, 11-16 July 2010.

deep syntactic information of an English sentence, which includes a fine-grained description of the syntactic property and a semantic representation of the sentence. We extract fine-grained translation rules from aligned HPSG tree/forest-string pairs. We localize an HPSG tree/forest to make it segmentable at any nodes to fit the extraction algorithms described in (Galley et al., 2006; Mi and Huang, 2008). We also propose a linear-time algorithm for extracting composed rules guided by predicate-argument structures. The effectiveness of the rules are testified in our tree-to-string and string-to-tree systems, taking bidirectional Japanese-English translations as our test cases.

This paper is organized as follows. In Section 2, we briefly review the tree-to-string and string-to-tree translation frameworks, tree-to-string rule extraction algorithms, and rich syntactic information previously used for SMT. The HPSG grammar and our proposal of fine-grained rule extraction algorithms are described in Section 3. Section 4 gives the experiments for applying fine-grained translation rules to large-scale Japanese-English translation tasks. Finally, we conclude in Section 5.

2 Related Work

2.1 Tree-to-string and string-to-tree translations

Tree-to-string translation (Liu et al., 2006; Huang et al., 2006) first uses a parser to parse a source sentence into a 1-best tree and then searches for the best derivation that segments and converts the tree into a target string. In contrast, string-to-tree translation (Galley et al., 2004; Galley et al., 2006; Chiang et al., 2009) is like *bilingual* parsing. That is, giving a (bilingual) translation grammar and a source sentence, we are trying to construct a parse forest in the target language. Consequently, the translation results can be collected from the leaves of the parse forest.

Figure 1 illustrates the training and decoding processes of bidirectional Japanese-English translations. The English sentence is "*John killed Mary*" and the Japanese sentence is "*jyon ha mari wo koroshita*", in which the function words "*ha*" and "*wo*" are not aligned with any English word.

2.2 Tree/forest-based rule extraction

Galley et al. (2004) proposed the GHKM algorithm for extracting (minimal) tree-to-string translation rules from a tuple of $\langle F, E_t, A\rangle$, where $F =$

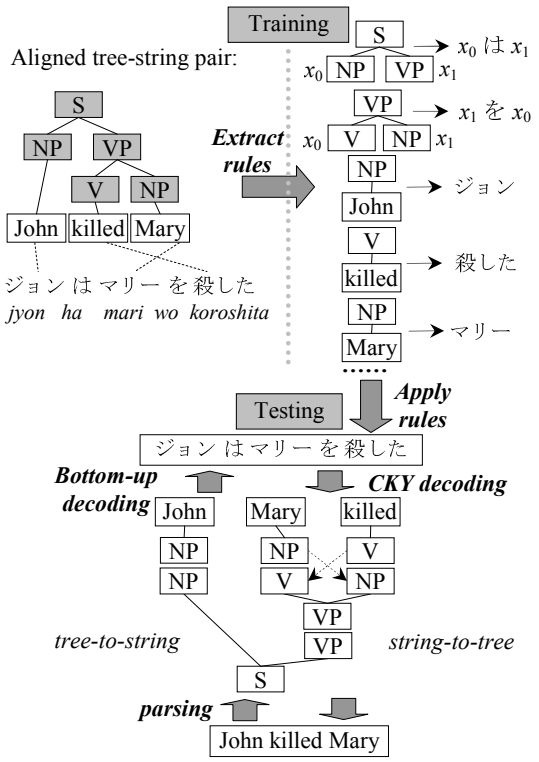


Figure 1: Illustration of the training and decoding processes for tree-to-string and string-to-tree translations.

f_1^J is a sentence of a foreign language other than English, E_t is a 1-best parse tree of an English sentence $E = e_1^I$, and $A = \{(j, i)\}$ is an alignment between the words in F and E.

The basic idea of GHKM algorithm is to decompose E_t into a series of tree fragments, each of which will form a rule with its corresponding translation in the foreign language. A is used as a constraint to guide the segmentation procedure, so that the root node of every tree fragment of E_t exactly corresponds to a *contiguous* span on the foreign language side. Based on this consideration, a *frontier set* (*fs*) is defined to be a set of nodes n in E_t that satisfies the following constraint:

$$fs = \{n | span(n) \cap comp\_span(n) = \phi\}. \quad (1)$$

Here, $span(n)$ is defined by the indices of the first and last word in F that are reachable from a node n, and $comp\_span(n)$ is defined to be the complement set of $span(n)$, i.e., the union of the spans of all nodes n' in E_t that are neither descendants nor ancestors of n. $span(n)$ and $comp\_span(n)$ of each n can be computed by first a bottom-up exploration and then a top-down traversal of E_t.

By restricting each fragment so that it only takes

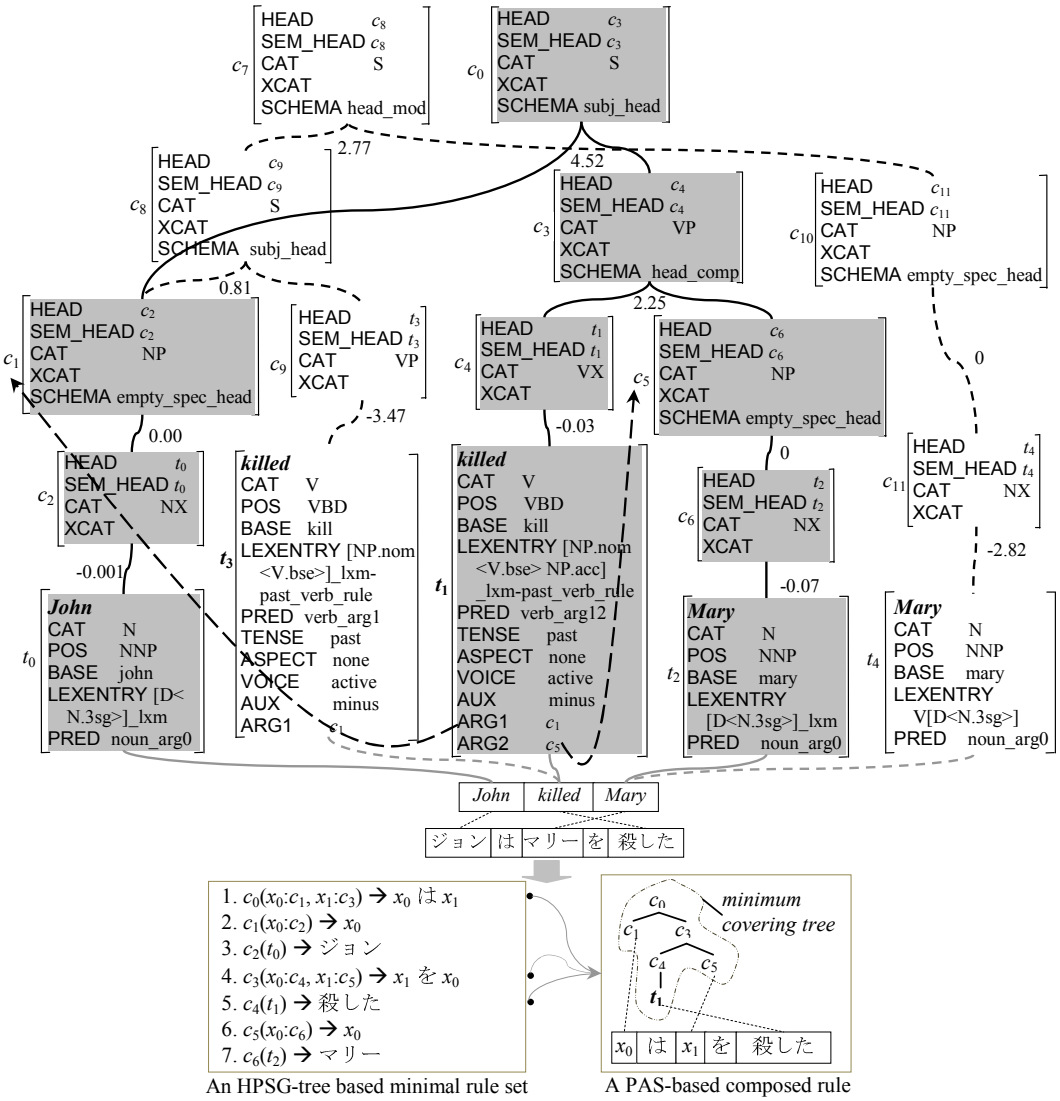


Figure 2: Illustration of an aligned HPSG forest-string pair. The forest includes two parse trees by taking "*Mary*" as a modifier (t_3, t_4) or an argument (t_1, t_2) of "*killed*". Arrows with broken lines denote the PAS dependencies from the terminal node t_1 to its argument nodes (c_1 and c_5). The scores of the hyperedges are attached to the forest as well.

the nodes in *fs* as the root and leaf nodes, a well-formed fragmentation of E_t is generated. With *fs* computed, rules are extracted through a depth-first traversal of E_t: we cut E_t at all nodes in *fs* to form tree fragments and extract a rule for each fragment. These extracted rules are called *minimal rules* (Galley et al., 2004). For example, the 1-best tree (with gray nodes) in Figure 2 is cut into 7 pieces, each of which corresponds to the tree fragment in a rule (bottom-left corner of the figure).

In order to include richer context information and account for multiple interpretations of unaligned words of foreign language, minimal rules which share adjacent tree fragments are connected together to form composed rules (Galley et al., 2006). For each aligned tree-string pair, Galley et al. (2006) constructed a derivation-forest, in which composed rules were generated, unaligned words of foreign language were consistently attached, and the translation probabilities of rules were estimated by using Expectation-Maximization (EM) (Dempster et al., 1977) training. For example, by combining the minimal rules of 1, 4, and 5, we obtain a composed rule, as shown in the bottom-right corner of Figure 2.

Considering the parse error problem in the 1-best or k-best parse trees, Mi and Huang (2008) extracted tree-to-string translation rules from aligned packed forest-string pairs. A forest compactly encodes exponentially many trees

rather than the 1-best tree used by Galley et al. (2004; 2006). Two problems were managed to be tackled during extracting rules from an aligned forest-string pair: where to cut and how to cut. Equation 1 was used again to compute a frontier node set to determine where to cut the packed forest into a number of tree-fragments. The difference with tree-based rule extraction is that the nodes in a packed forest (which is a hypergraph) now are hypernodes, which can take a set of incoming hyperedges. Then, by limiting each fragment to be a tree and whose root/leaf hypernodes all appearing in the frontier set, the packed forest can be segmented properly into a set of tree fragments, each of which can be used to generate a tree-to-string translation rule.

2.3 Rich syntactic information for SMT

Before describing our approaches of applying deep syntactic information yielded by an HPSG parser for fine-grained rule extraction, we would like to briefly review what kinds of deep syntactic information have been employed for SMT.

Two kinds of supertags, from Lexicalized Tree-Adjoining Grammar and Combinatory Categorial Grammar (CCG), have been used as lexical syntactic descriptions (Hassan et al., 2007) for phrase-based SMT (Koehn et al., 2007). By introducing supertags into the target language side, i.e., the target language model and the target side of the phrase table, significant improvement was achieved for Arabic-to-English translation. Birch et al. (2007) also reported a significant improvement for Dutch-English translation by applying CCG supertags at a word level to a factorized SMT system (Koehn et al., 2007).

In this paper, we also make use of supertags on the English language side. In an HPSG parse tree, these lexical syntactic descriptions are included in the LEXENTRY feature (refer to Table 2) of a lexical node (Matsuzaki et al., 2007). For example, the LEXENTRY feature of "t_1*:killed*" takes the value of `[NP.nom<V.bse>NP.acc]_lxm-past_verb_rule` in Figure 2. In which, `[NP.nom<V.bse>NP.acc]` is an HPSG style supertag, which tells us that the base form of "*killed*" needs a nominative NP in the left hand side and an accessorial NP in the right hand side. The major differences are that, we use a larger feature set (Table 2) including the supertags for fine-grained tree-to-string rule extraction, rather than string-to-string translation (Hassan et al., 2007; Birch et al., 2007).

The Logon project[2] (Oepen et al., 2007) for Norwegian-English translation integrates in-depth grammatical analysis of Norwegian (using lexical functional grammar, similar to (Riezler and Maxwell, 2006)) with semantic representations in the minimal recursion semantics framework, and fully grammar-based generation for English using HPSG. A hybrid (of rule-based and data-driven) architecture with a semantic transfer backbone is taken as the vantage point of this project. In contrast, the fine-grained tree-to-string translation rule extraction approaches in this paper are totally data-driven, and easily applicable to numerous language pairs by taking English as the source or target language.

3 Fine-grained rule extraction

We now introduce the deep syntactic information generated by an HPSG parser and then describe our approaches for fine-grained tree-to-string rule extraction. Especially, we localize an HPSG tree/forest to fit the extraction algorithms described in (Galley et al., 2006; Mi and Huang, 2008). Also, we propose a linear-time composed rule extraction algorithm by making use of predicate-argument structures.

3.1 Deep syntactic information by HPSG parsing

Head-driven phrase structure grammar (HPSG) is a lexicalist grammar framework. In HPSG, linguistic entities such as words and phrases are represented by a data structure called a *sign*. A sign gives a factored representation of the syntactic features of a word/phrase, as well as a representation of their semantic content. Phrases and words represented by signs are composed into larger phrases by applications of *schemata*. The semantic representation of the new phrase is calculated at the same time. As such, an HPSG parse tree/forest can be considered as a tree/forest of signs (c.f. the HPSG forest in Figure 2).

An HPSG parse tree/forest has two attractive properties as a representation of an English sentence in syntax-based SMT. First, we can carefully control the condition of the application of a translation rule by exploiting the fine-grained syntactic

[2] http://www.emmtee.net/

Feature	Description
CAT	phrasal category
XCAT	fine-grained phrasal category
SCHEMA	name of the schema applied in the node
HEAD	*pointer* to the head daughter
SEM_HEAD	*pointer* to the semantic head daughter
CAT	syntactic category
POS	Penn Treebank-style part-of-speech tag
BASE	base form
TENSE	tense of a verb (past, present, untensed)
ASPECT	aspect of a verb (none, perfect, progressive, perfect-progressive)
VOICE	voice of a verb (passive, active)
AUX	auxiliary verb or not (minus, modal, have, be, do, to, copular)
LEXENTRY	lexical entry, with supertags embedded
PRED	type of a predicate
ARG$\langle x \rangle$	*pointer* to semantic arguments, $x = 1..4$

Table 2: Syntactic/semantic features extracted from HPSG signs that are included in the output of Enju. Features in phrasal nodes (top) and lexical nodes (bottom) are listed separately.

description in the English parse tree/forest, as well as those in the translation rules. Second, we can identify sub-trees in a parse tree/forest that correspond to basic units of the semantics, namely sub-trees covering a predicate and its arguments, by using the semantic representation given in the signs. We expect that extraction of translation rules based on such *semantically-connected* subtrees will give a compact and effective set of translation rules.

A sign in the HPSG tree/forest is represented by a typed feature structure (TFS) (Carpenter, 1992). A TFS is a directed-acyclic graph (DAG) wherein the edges are labeled with feature names and the nodes (feature values) are typed. In the original HPSG formalism, the types are defined in a hierarchy and the DAG can have arbitrary shape (e.g., it can be of any depth). We however use a simplified form of TFS, for simplicity of the algorithms. In the simplified form, a TFS is converted to a (flat) set of pairs of feature names and their values. Table 2 lists the features used in this paper, which are a subset of those in the original output from an HPSG parser, Enju[3]. The HPSG forest shown in Figure 2 is in this simplified format. An important detail is that we allow a feature value to be a pointer to another (simplified) TFS. Such pointer-valued features are necessary for denoting the semantics, as explained shortly.

In the Enju English HPSG grammar (Miyao et al., 2003) used in this paper, the semantic content of a sentence/phrase is represented by a predicate-argument structure (PAS). Figure 3 shows the PAS of the example sentence in Figure 2, "*John killed Mary*", and a more complex PAS for another sentence, "*She ignored the fact that I wanted to dispute*", which is adopted from (Miyao et al., 2003). In an HPSG tree/forest, each leaf node generally introduces a predicate, which is represented by the pair of LEXENTRY (lexical entry) feature and PRED (predicate type) feature. The arguments of a predicate are designated by the pointers from the ARG$\langle x \rangle$ features in a leaf node to non-terminal nodes.

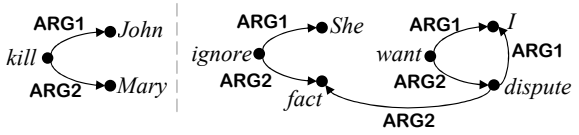


Figure 3: Predicate argument structures for the sentences of "*John killed Mary*" and "*She ignored the fact that I wanted to dispute*".

3.2 Localize HPSG forest

Our fine-grained translation rule extraction algorithm is sketched in Algorithm 1. Considering that a parse tree is a trivial packed forest, we only use the term forest to expand our discussion, hereafter. Recall that there are pointer-valued features in the TFSs (Table 2) which prevent arbitrary segmentation of a packed forest. Hence, we have to localize an HPSG forest.

For example, there are ARG pointers from t_1 to c_1 and c_5 in the HPSG forest of Figure 2. However, the three nodes are not included in one (minimal) translation rule. This problem is caused by not considering the predicate argument dependency among t_1, c_1, and c_5 while performing the GHKM algorithm. We can combine several minimal rules (Galley et al., 2006) together to address this dependency. Yet we have a faster way to tackle PASs, as will be described in the next subsection.

Even if we omit ARG, there are still two kinds of pointer-valued features in TFSs, HEAD and SEM_HEAD. Localizing these pointer-valued features is straightforward, since during parsing, the HEAD and SEM_HEAD of a node are automatically transferred to its mother node. That is, the syntactic and semantic head of a node only take

[3]http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

Algorithm 1 Fine-grained rule extraction

Input: HPSG tree/forest E_f, foreign sentence F, and alignment A
Output: a PAS-based rule set $\mathcal{R}_1$ and/or a tree-rule set $\mathcal{R}_2$

1: **if** E_f is an HPSG tree **then**
2: $E_f^{'}$ = localize_Tree(E_f)
3: $\mathcal{R}_1$ = PASR_extraction($E_f^{'}$, F, A) ▷ Algorithm 2
4: $E_f^{''}$ = ignore_PAS($E_f^{'}$)
5: $\mathcal{R}_2$ = TR_extraction($E_f^{''}$, F, A) ▷ composed rule extraction algorithm in (Galley et al., 2006)
6: **else if** E_f is an HPSG forest **then**
7: $E_f^{'}$ = localize_Forest(E_f);
8: $\mathcal{R}_2$ = forest_based_rule_extraction($E_f^{'}$, F, A) ▷ Algorithm 1 in (Mi and Huang, 2008)
9: **end if**

the identifier of the daughter node as the values. For example, HEAD and SEM_HEAD of node c_0 take the identical value to be c_3 in Figure 2.

To extract tree-to-string rules from the tree structures of an HPSG forest, our solution is to pre-process an HPSG forest in the following way:

- for a phrasal hypernode, replace its HEAD and SEM_HEAD value with L, R, or S, which respectively represent left daughter, right daughter, or single daughter (line 2 and 7); and,
- for a lexical node, ARG⟨x⟩ and PRED features are ignored (line 4).

A pure syntactic-based HPSG forest without any pointer-valued features can be yielded through this pre-processing for the consequent execution of the extraction algorithms (Galley et al., 2006; Mi and Huang, 2008).

3.3 Predicate-argument structures

In order to extract translation rules from PASs, we want to localize a predicate word and its arguments into one tree fragment. For example, in Figure 2, we can use a tree fragment which takes c_0 as its root node and c_1, t_1, and c_5 on its yield (= leaf nodes of a tree fragment) to cover "*killed*" and its subject and direct object arguments. We define this kind of tree fragment to be a *minimum covering tree*. For example, the minimum covering tree of $\{t_1, c_1, c_5\}$ is shown in the bottom-right corner of Figure 2. The definition supplies us a linear-time algorithm to directly find the tree fragment that covers a PAS during both rule extracting and rule matching when decoding an HPSG tree.

Algorithm 2 PASR extraction

Input: HPSG tree E_t, foreign sentence F, and alignment A
Output: a PAS-based rule set $\mathcal{R}$

1: $\mathcal{R}$ = {}
2: **for** node $n \in$ Leaves(E_t) **do**
3: **if** Open(n.ARG) **then**
4: T_c = MinimumCoveringTree(E_t, n, n.ARGs)
5: **if** root and leaf nodes of T_c are in *fs* **then**
6: generate a rule r using fragment T_c
7: $\mathcal{R}$.append(r)
8: **end if**
9: **end if**
10: **end for**

See (Wu, 2010) for more examples of minimum covering trees.

Taking a minimum covering tree as the tree fragment, we can easily build a tree-to-string translation rule that reflects the semantic dependency of a PAS. The algorithm of PAS-based rule (PASR) extraction is sketched in Algorithm 2. Suppose we are given a tuple of $\langle F, E_t, A\rangle$. E_t is pre-processed by replacing HEAD and SEM_HEAD to be L, R, or S, and computing the $span$ and $comp\_span$ of each node.

We extract PAS-based rules through one-time traversal of the leaf nodes in E_t (line 2). For each leaf node n, we extract a minimum covering tree T_c if n contains at least one argument. That is, at least one ARG⟨x⟩ takes the value of some node identifier, where x ranges 1 over 4 (line 3). Then, we require the root and yield nodes of T_c being in the frontier set of E_t (line 5). Based on T_c, we can easily build a tree-to-string translation rule by further completing the right-hand-side string by sorting the spans of T_c's leaf nodes, lexicalizing the terminal node's span(s), and assigning a variable to each non-terminal node's span. Maximum likelihood estimation is used to calculate the translation probabilities of each rule.

An example of PAS-based rule is shown in the bottom-right corner of Figure 2. In the rule, the subject and direct-object of "*killed*" are generalized into two variables, x_0 and x_1.

4 Experiments

4.1 Translation models

We use a tree-to-string model and a string-to-tree model for bidirectional Japanese-English translations. Both models use a phrase translation table (PTT), an HPSG tree-based rule set (TRS), and a PAS-based rule set (PRS). Since the three rule sets are independently extracted and estimated, we

use Minimum Error Rate Training (MERT) (Och, 2003) to tune the weights of the features from the three rule sets on the development set.

Given a 1-best (localized) HPSG tree E_t, the tree-to-string decoder searches for the optimal derivation d^* that transforms E_t into a Japanese string among the set of all possible derivations D:

$$d^* = \arg\max_{d \in D} \{\lambda_1 \log p_{LM}(\tau(d)) + \lambda_2 |\tau(d)| + \log s(d|E_t)\}. \quad (2)$$

Here, the first item is the language model (LM) probability where $\tau(d)$ is the target string of derivation d; the second item is the translation length penalty; and the third item is the translation score, which is decomposed into a product of feature values of rules:

$$s(d|E_t) = \prod_{r \in d} f(r_{\in PTT}) f(r_{\in TRS}) f(r_{\in PRS}).$$

This equation reflects that the translation rules in one d come from three sets. Inspired by (Liu et al., 2009b), it is appealing to combine these rule sets together in one decoder because PTT provides excellent rule coverages while TRS and PRS offer linguistically motivated phrase selections and non-local reorderings. Each $f(r)$ is in turn a product of five features:

$$f(r) = p(s|t)^{\lambda_3} \cdot p(t|s)^{\lambda_4} \cdot l(s|t)^{\lambda_5} \cdot l(t|s)^{\lambda_6} \cdot e^{\lambda_7}.$$

Here, s/t represent the source/target part of a rule in PTT, TRS, or PRS; $p(\cdot|\cdot)$ and $l(\cdot|\cdot)$ are translation probabilities and lexical weights of rules from PTT, TRS, and PRS. The derivation length penalty is controlled by λ_7.

In our string-to-tree model, for efficient decoding with integrated n-gram LM, we follow (Zhang et al., 2006) and inversely binarize all translation rules into Chomsky Normal Forms that contain at most two variables and can be incrementally scored by LM. In order to make use of the binarized rules in the CKY decoding, we add two kinds of glues rules:

$$\begin{aligned} S &\rightarrow {X_m}^{(1)}, {X_m}^{(1)}; \\ S &\rightarrow S^{(1)} {X_m}^{(2)}, S^{(1)} {X_m}^{(2)}. \end{aligned}$$

Here X_m ranges over the nonterminals appearing in a binarized rule set. These glue rules can be seen as an extension from X to $\{X_m\}$of the two glue rules described in (Chiang, 2007).

The string-to-tree decoder searches for the optimal derivation d^* that parses a Japanese string F into a packed forest of the set of all possible derivations D:

$$d^* = \arg\max_{d \in D} \{\lambda_1 \log p_{LM}(\tau(d)) + \lambda_2 |\tau(d)| + \lambda_3 g(d) + \log s(d|F)\}. \quad (3)$$

This formula differs from Equation 2 by replacing E_t with F in $s(d|\cdot)$ and adding $g(d)$, which is the number of glue rules used in d. Further definitions of $s(d|F)$ and $f(r)$ are identical with those used in Equation 2.

4.2 Decoding algorithms

In our translation models, we have made use of three kinds of translation rule sets which are trained separately. We perform *derivation-level combination* as described in (Liu et al., 2009b) for mixing different types of translation rules within one derivation.

For tree-to-string translation, we use a bottom-up beam search algorithm (Liu et al., 2006) for decoding an HPSG tree E_t. We keep at most 10 best derivations with distinct $\tau(d)$s at each node.

Recall the definition of minimum covering tree, which supports a faster way to retrieve available rules from PRS without generating all the subtrees. That is, when node n fortunately to be the root of some minimum covering tree(s), we use the tree(s) to seek available PAS-based rules in PRS. We keep a hash-table with the key to be the node identifier of n and the value to be a priority queue of available PAS-based rules. The hash-table is easy to be filled by one-time traversal of the terminal nodes in E_t. At each terminal node, we seek its minimum covering tree, retrieve PRS, and update the hash-table. For example, suppose we are decoding an HPSG tree (with gray nodes) shown in Figure 2. At t_1, we can extract its minimum covering tree with the root node to be c_0, then take this tree fragment as the key to retrieve PRS, and consequently put c_0 and the available rules in the hash-table. When decoding at c_0, we can directly access the hash-table looking for available PAS-based rules.

In contrast, we use a CKY-style algorithm with beam-pruning and cube-pruning (Chiang, 2007) to decode Japanese sentences. For each Japanese sentence F, the output of the chart-parsing algorithm is expressed as a hypergraph representing a set of derivations. Given such a hypergraph, we

	Train	Dev.	Test
# of sentences	994K	2K	2K
# of Jp words	28.2M	57.4K	57.1K
# of En words	24.7M	50.3K	49.9K

Table 3: Statistics of the JST corpus.

use the Algorithm 3 described in (Huang and Chiang, 2005) to extract its k-best ($k = 500$ in our experiments) derivations. Since different derivations may lead to the same target language string, we further adopt Algorithm 3's modification, i.e., keep a hash-table to maintain the unique target sentences (Huang et al., 2006), to efficiently generate the unique k-best translations.

4.3 Setups

The JST Japanese-English paper abstract corpus[4], which consists of one million parallel sentences, was used for training and testing. This corpus was constructed from a Japanese-English paper abstract corpus by using the method of Utiyama and Isahara (2007). Table 3 shows the statistics of this corpus. Making use of Enju 2.3.1, we successfully parsed 987,401 English sentences in the training set, with a parse rate of 99.3%. We modified this parser to output a packed forest for each English sentence.

We executed GIZA++ (Och and Ney, 2003) and *grow-diag-final-and* balancing strategy (Koehn et al., 2007) on the training set to obtain a phrase-aligned parallel corpus, from which bidirectional phrase translation tables were estimated. SRI Language Modeling Toolkit (Stolcke, 2002) was employed to train 5-gram English and Japanese LMs on the training set. We evaluated the translation quality using the case-insensitive BLEU-4 metric (Papineni et al., 2002). The MERT toolkit we used is Z-mert[5] (Zaidan, 2009).

The baseline system for comparison is Joshua (Li et al., 2009), a freely available decoder for hierarchical phrase-based SMT (Chiang, 2005). We respectively extracted 4.5M and 5.3M translation rules from the training set for the 4K English and Japanese sentences in the development and test sets. We used the default configuration of Joshua, expect setting the maximum number of items/rules and the k of k-best outputs to be the identical 200 for English-to-Japanese translation and 500 for Japanese-to-English translation.

We used four dual core Xeon machines (4×3.0GHz×2CPU, 4×64GB memory) to run all the experiments.

[4]http://www.jst.go.jp. The corpus can be conditionally obtained from NTCIR-7 patent translation workshop homepage: http://research.nii.ac.jp/ntcir/permission/ntcir-7/perm-en-PATMT.html.

[5]http://www.cs.jhu.edu/ ozaidan/zmert/

	PRS	C_3^S	C_3	F^S	F
tree nodes	TFS	POS	TFS	POS	TFS
# rules	0.9	62.1	83.9	92.5	103.7
# tree types	0.4	23.5	34.7	40.6	45.2
extract time	3.5	-	98.6	-	121.2

Table 4: Statistics of several kinds of tree-to-string rules. Here, the number is in million level and the time is in hour.

4.4 Results

Table 4 illustrates the statistics of several translation rule sets, which are classified by:

- using TFSs or simple POS/phrasal tags (annotated by a superscript S) to represent tree nodes;
- composed rules (PRS) extracted from the PAS of 1-best HPSG trees;
- composed rules (C_3), extracted from the tree structures of 1-best HPSG trees, and 3 is the maximum number of internal nodes in the tree fragments; and
- forest-based rules (F), where the packed forests are pre-pruned by the marginal probability-based inside-outside algorithm used in (Mi and Huang, 2008).

Table 5 reports the BLEU-4 scores achieved by decoding the test set making use of Joshua and our systems (t2s = tree-to-string and s2t = string-to-tree) under numerous rule sets. We analyze this table in terms of several aspects to prove the effectiveness of deep syntactic information for SMT.

Let's first look at the performance of TFSs. We take C_3^S and F^S as approximations of CFG-based translation rules. Comparing the BLEU-4 scores of PTT+C_3^S and PTT+C_3, we gained 0.56 (t2s) and 0.57 (s2t) BLEU-4 points which are significant improvements ($p < 0.05$). Furthermore, we gained 0.50 (t2s) and 0.62 (s2t) BLEU-4 points from PTT+F^S to PTT+F, which are also significant improvements ($p < 0.05$). The rich features included in TFSs contribute to these improvements.

Systems	BLEU-t2s	Decoding	BLEU-s2t
Joshua	21.79	0.486	19.73
PTT	18.40	0.013	17.21
PTT+PRS	22.12	0.031	19.33
PTT+C_3^S	23.56	2.686	20.59
PTT+C_3	24.12	2.753	21.16
PTT+C_3+PRS	24.13	2.930	21.20
PTT+F^S	24.25	3.241	22.05
PTT+F	**24.75**	3.470	**22.67**

Table 5: BLEU-4 scores (%) achieved by Joshua and our systems under numerous rule configurations. The decoding time (seconds per sentence) of tree-to-string translation is listed as well.

Also, BLEU-4 scores were inspiringly increased 3.72 (t2s) and 2.12 (s2t) points by appending PRS to PTT, comparing PTT with PTT+PRS. Furthermore, in Table 5, the decoding time (seconds per sentence) of tree-to-string translation by using PTT+PRS is more than 86 times faster than using the other tree-to-string rule sets. This suggests that the direct generation of minimum covering trees for rule matching is extremely faster than generating all subtrees of a tree node. Note that PTT performed extremely bad compared with all other systems or tree-based rule sets. The major reason is that we did not perform any reordering or distorting during decoding with PTT.

However, in both t2s and s2t systems, the BLEU-4 score benefits of PRS were covered by the composed rules: both PTT+C_3^S and PTT+C_3 performed significant better ($p < 0.01$) than PTT+PRS, and there are no significant differences when appending PRS to PTT+C_3. The reason is obvious: PRS is only a small subset of the composed rules, and the probabilities of rules in PRS were estimated by maximum likelihood, which is fast but biased compared with EM based estimation (Galley et al., 2006).

Finally, by using PTT+F, our systems achieved the best BLEU-4 scores of 24.75% (t2s) and 22.67% (s2t), both are significantly better ($p < 0.01$) than that achieved by Joshua.

5 Conclusion

We have proposed approaches of using deep syntactic information for extracting fine-grained tree-to-string translation rules from aligned HPSG forest-string pairs. The main contributions are the applications of GHKM-related algorithms (Galley et al., 2006; Mi and Huang, 2008) to HPSG forests and a linear-time algorithm for extracting composed rules from predicate-argument structures. We applied our fine-grained translation rules to a tree-to-string system and an Hiero-style string-to-tree system. Extensive experiments on large-scale bidirectional Japanese-English translations testified the significant improvements on BLEU score.

We argue the fine-grained translation rules are generic and applicable to many syntax-based SMT frameworks such as the forest-to-string model (Mi et al., 2008). Furthermore, it will be interesting to extract fine-grained tree-to-tree translation rules by integrating deep syntactic information in the source and/or target language side(s). These tree-to-tree rules are applicable for forest-to-tree translation models (Liu et al., 2009a).

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Japanese/Chinese Machine Translation Project in Special Coordination Funds for Promoting Science and Technology (MEXT, Japan), and Microsoft Research Asia Machine Translation Theme. The first author thanks Naoaki Okazaki and Yusuke Miyao for their help and the anonymous reviewers for improving the earlier version.

References

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. Ccg supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16, June.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 218–226, June.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, MI.

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, Sydney.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of ACL*, pages 288–295, June.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of 7th AMTA*.

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

Zhifei Li, Chris Callison-Burch, Chris Dyery, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Demonstration of joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 25–28, August.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment templates for statistical machine transaltion. In *Proceedings of COLING-ACL*, pages 609–616, Sydney, Australia.

Yang Liu, Yajuan Lü, and Qun Liu. 2009a. Improving tree-to-tree translation with packed forests. In *Proceedings of ACL-IJCNLP*, pages 558–566, August.

Yang Liu, Haitao Mi, Yang Feng, and Qun Liu. 2009b. Joint decoding with multiple translation models. In *Proceedings of ACL-IJCNLP*, pages 576–584, August.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient hpsg parsing with supertagging and cfg-filtering. In *Proceedings of IJCAI*, pages 1671–1676, January.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, October.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08:HLT*, pages 192–199, Columbus, Ohio.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 285–291, Borovets.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Stephan Oepen, Erik Velldal, Jan Tore Lønning, Paul Meurer, and Victoria Rosén. 2007. Towards hybrid quality-oriented machine translation - on linguistics and probabilities in mt. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-07)*, September.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Stefan Riezler and John T. Maxwell, III. 2006. Grammatical machine translation. In *Proceedings of HLT-NAACL*, pages 248–255.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, pages 901–904.

Masao Utiyama and Hitoshi Isahara. 2007. A japanese-english patent parallel corpus. In *Proceedings of MT Summit XI*, pages 475–482, Copenhagen.

Xianchao Wu. 2010. *Statistical Machine Translation Using Large-Scale Lexicon and Deep Syntactic Structures*. Ph.D. dissertation. Department of Computer Science, The University of Tokyo.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*, pages 256–263, June.

Accurate Context-Free Parsing with Combinatory Categorial Grammar

Timothy A. D. Fowler and **Gerald Penn**
Department of Computer Science, University of Toronto
Toronto, ON, M5S 3G4, Canada
{tfowler, gpenn}@cs.toronto.edu

Abstract

The definition of combinatory categorial grammar (CCG) in the literature varies quite a bit from author to author. However, the differences between the definitions are important in terms of the language classes of each CCG. We prove that a wide range of CCGs are strongly context-free, including the CCG of CCGbank and of the parser of Clark and Curran (2007). In light of these new results, we train the PCFG parser of Petrov and Klein (2007) on CCGbank and achieve state of the art results in supertagging accuracy, PARSEVAL measures and dependency accuracy.

1 Introduction

Combinatory categorial grammar (CCG) is a variant of categorial grammar which has attracted interest for both theoretical and practical reasons. On the theoretical side, we know that it is mildly context-sensitive (Vijay-Shanker and Weir, 1994) and that it can elegantly analyze a wide range of linguistic phenomena (Steedman, 2000). On the practical side, we have corpora with CCG derivations for each sentence (Hockenmaier and Steedman, 2007), a wide-coverage parser trained on that corpus (Clark and Curran, 2007) and a system for converting CCG derivations into semantic representations (Bos et al., 2004).

However, despite being treated as a single unified grammar formalism, each of these authors use variations of CCG which differ primarily on which combinators are included in the grammar and the restrictions that are put on them. These differences are important because they affect whether the mild context-sensitivity proof of Vijay-Shanker and Weir (1994) applies. We will provide a generalized framework for CCG within which the full variation of CCG seen in the literature can be defined. Then, we prove that for a wide range of CCGs there is a context-free grammar (CFG) that has exactly the same derivations. Included in this class of strongly context-free CCGs are a grammar including all the derivations in CCGbank and the grammar used in the Clark and Curran parser.

Due to this insight, we investigate the potential of using tools from the probabilistic CFG community to improve CCG parsing results. The Petrov parser (Petrov and Klein, 2007) uses latent variables to refine the grammar extracted from a corpus to improve accuracy, originally used to improve parsing results on the Penn treebank (PTB). We train the Petrov parser on CCGbank and achieve the best results to date on sentences from section 23 in terms of supertagging accuracy, PARSEVAL measures and dependency accuracy.

These results should not be interpreted as proof that grammars extracted from the Penn treebank and from CCGbank are equivalent. Bos's system for building semantic representations from CCG derivations is only possible due to the categorial nature of CCG. Furthermore, the long distance dependencies involved in extraction and coordination phenomena have a more natural representation in CCG.

2 The Language Classes of Combinatory Categorial Grammars

A categorial grammar is a grammatical system consisting of a finite set of words, a set of categories, a finite set of sentential categories, a finite lexicon mapping words to categories and a rule system dictating how the categories can be combined. The set of categories are constructed from a finite set of atoms A (e.g. $A = \{S, NP, N, PP\}$) and a finite set of binary connectives B (e.g. $B = \{/, \backslash\}$) to build an infinite set of categories $\mathcal{C}(A, B)$ (e.g. $\mathcal{C}(A, B) = \{S, S\backslash NP, (S\backslash NP)/NP, \ldots\}$). For a category C, its size $|C|$ is the

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 335–344,
Uppsala, Sweden, 11-16 July 2010.

number of atom occurrences it contains. When not specified, connectives are left associative.

According to the literature, combinatory categorial grammar has been defined to have a variety of rule systems. These rule systems vary from a small rule set, motivated theoretically (Vijay-Shanker and Weir, 1994), to a larger rule set, motivated linguistically, (Steedman, 2000) to a very large rule set, motivated by practical coverage (Hockenmaier and Steedman, 2007; Clark and Curran, 2007). We provide a definition general enough to incorporate these four main variants of CCG, as well as others.

A combinatory categorial grammar (CCG) is a categorial grammar whose rule system consists of rule schemata where the left side is a sequence of categories and the right side is a single category where the categories may include variables over both categories and connectives. In addition, rule schemata may specify a sequence of categories and connectives using the ... convention[1]. When ... appears in a rule, it matches any sequence of categories and connectives according to the connectives adjacent to the .... For example, the rule schema for forward composition is:

$$X/Y, Y/Z \rightarrow X/Z$$

and the rule schema for generalized forward crossed composition is:

$$X/Y, Y|_1Z_1|_2 \ldots |_nZ_n \rightarrow X|_1Z_1|_2 \ldots |_nZ_n$$

where X, Y and Z_i for $1 \leq i \leq n$ are variables over categories and $|_i$ for $1 \leq i \leq n$ are variables over connectives. Figure 1 shows a CCG derivation from CCGbank.

A well-known categorial grammar which is not a CCG is Lambek categorial grammar (Lambek, 1958) whose introduction rules cannot be characterized as combinatory rules (Zielonka, 1981).

2.1 Classes for defining CCG

We define a number of *schema classes* general enough that the important variants of CCG can be defined by selecting some subset of the classes. In addition to the schema classes, we also define two *restriction classes* which define ways in which the rule schemata from the schema classes can be restricted. We define the following schema classes:

(1) Application

- $X/Y, Y \rightarrow X$
- $Y, X\backslash Y \rightarrow X$

(2) Composition

- $X/Y, Y/Z \rightarrow X/Z$
- $Y\backslash Z, X\backslash Y \rightarrow X\backslash Z$

(3) Crossed Composition

- $X/Y, Y\backslash Z \rightarrow X\backslash Z$
- $Y/Z, X\backslash Y \rightarrow X/Z$

(4) Generalized Composition

- $X/Y, Y/Z_1/\ldots/Z_n \rightarrow X/Z_1/\ldots/Z_n$
- $Y\backslash Z_1\backslash\ldots\backslash Z_n, X\backslash Y \rightarrow X\backslash Z_1\backslash\ldots\backslash Z_n$

(5) Generalized Crossed Composition

- $X/Y, Y|_1Z_1|_2\ldots|_nZ_n \rightarrow X|_1Z_1|_2\ldots|_nZ_n$
- $Y|_1Z_1|_2\ldots|_nZ_n, X\backslash Y \rightarrow X|_1Z_1|_2\ldots|_nZ_n$

(6) Reducing Generalized Crossed Composition

Generalized Composition or Generalized Crossed Composition where $|X| \leq |Y|$.

(7) Substitution

- $(X/Y)|_1Z, Y|_1Z \rightarrow X|_1Z$
- $Y|_1Z, (X\backslash Y)|_1Z \rightarrow X|_1Z$

(8) D Combinator[2]

- $X/(Y|_1Z), Y|_2W \rightarrow X|_2(W|_1Z)$
- $Y|_2W, X\backslash(Y|_1Z) \rightarrow X|_2(W|_1Z)$

(9) Type-Raising

- $X \rightarrow T/(T\backslash X)$
- $X \rightarrow T\backslash(T/X)$

(10) Finitely Restricted Type-Raising

- $X \rightarrow T/(T\backslash X)$ where $\langle X, T\rangle \in S$ for finite S
- $X \rightarrow T\backslash(T/X)$ where $\langle X, T\rangle \in S$ for finite S

(11) Finite Unrestricted Variable-Free Rules

- $\vec{X} \rightarrow Y$ where $\langle \vec{X}, Y\rangle \in S$ for finite S

[1]The ... convention (Vijay-Shanker and Weir, 1994) is essentially identical to the $ convention of Steedman (2000).

[2]Hoyt and Baldridge (2008) argue for the inclusion of the D Combinator in CCG.

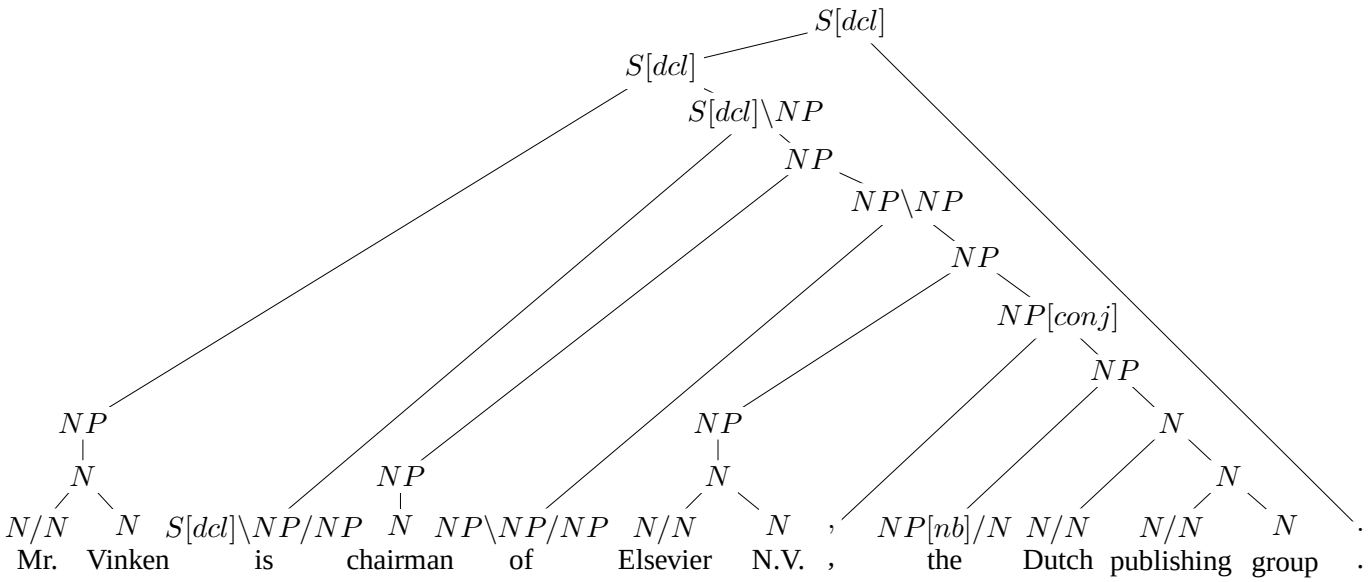


Figure 1: A CCG derivation from section 00 of CCGbank.

We define the following restriction classes:

(A) Rule Restriction to a Finite Set

The rule schemata in the schema classes of a CCG are limited to a finite number of instantiations.

(B) Rule Restrictions to Certain Categories [3]

The rule schemata in the schema classes of a CCG are limited to a finite number of instantiations although variables are allowed in the instantiations.

Vijay-Shanker and Weir (1994) define CCG to be schema class (4) with restriction class (B). Steedman (2000) defines CCG to be schema classes (1-5), (6), (10) with restriction class (B).

2.2 Strongly Context-Free CCGs

Proposition 1. *The set of atoms in any derivation of any CCG consisting of a subset of the schema classes (1-8) and (10-11) is finite.*

Proof. A finite lexicon can introduce only a finite number of atoms in lexical categories.

Any rule corresponding to a schema in the schema classes (1-8) has only those atoms on the right that occur somewhere on the left. Rules in classes (10-11) can each introduce a finite number of atoms, but there can be only a finite number of such rules, limiting the new atoms to a finite number. □

Definition 1. The *subcategories* for a category c are c_1 and c_2 if $c = c_1 \bullet c_2$ for $\bullet \in B$ and c if c is atomic. Its *second subcategories* are the subcategories of its subcategories.

Proposition 2. *Any CCG consisting of a subset of the rule schemata (1-3), (6-8) and (10-11) has derivations consisting of only a finite number of categories.*

Proof. We first prove the proposition excluding schema class (8). We will use structural induction on the derivations to prove that there is a bound on the size of the subcategories of any category in the derivation. The base case is the assignment of a lexical category to a word and the inductive step is the use of a rule from schema classes (1-4), (6-7) and (10-11).

Given that the lexicon is finite, there is a bound k on the size of the subcategories of lexical categories. Furthermore, there is a bound l on the size of the subcategories of categories on the right side of any rule in (10) and (11). Let $m = max(k, l)$.

For rules from schema class (1), the category on the right is a subcategory of the first category on the left, so the subcategories on the right are bound by m. For rules from schema classes (2-3), the category on the right has subcategories X and Z each of which is bound in size by m since they occur as subcategories of categories on the left.

For rules from schema class (6), since reducing generalized composition is a special case of re-

[3] Baldridge (2002) introduced a variant of CCG where modalities are added to the connectives / and \ along with variants of the combinatory rules based on these modalities. Our proofs about restriction class (B) are essentially identical to proofs regarding the multi-modal variant.

ducing generalized crossing composition, we need only consider the latter. The category on the right has subcategories $X|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ and Z_n. Z_n is bound in size by m because it occurs as a subcategory of the second category on the left. Then, the size of $Y|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ must be bound by m and since $|X| \leq |Y|$, the size of $X|_1Z_1|_2 \ldots |_{n-1}|Z_{n-1}$ must also be bound by m.

For rules from schema class (7), the category on the right has subcategories X and Z. The size of Z is bound by m because it is a subcategory of a category on the left. The size of X is bound by m because it is a second subcategory of a category on the left.

Finally, the use of rules in schema classes (10-11) have categories on the right that are bounded by l, which is, in turn, bounded by m. Then, by proposition 1, there must only be a finite number of categories in any derivation in a CCG consisting of a subset of rule schemata (1-3), (6-7) and (10-11).

The proof including schema class (8) is essentially identical except that k must be defined in terms of the size of the second subcategories. □

Definition 2. A grammar is *strongly context-free* if there exists a CFG such that the derivations of the two grammars are identical.

Proposition 3. *Any CCG consisting of a subset of the schema classes (1-3), (6-8) and (10-11) is strongly context-free.*

Proof. Since the CCG generates derivations whose categories are finite in number let C be that set of categories. Let $S(C, X)$ be the subset of C matching category X (which may have variables). Then, for each rule schema $C_1, C_2 \rightarrow C_3$ in (1-3) and (6-8), we construct a context-free rule $C'_3 \rightarrow C'_1, C'_2$ for each C'_i in $S(C, C_i)$ for $1 \leq i \leq 3$. Similarly, for each rule schema $C_1 \rightarrow C_2$ in (10), we construct a context-free rule $C'_2 \rightarrow C'_1$ which results in a finite number of such rules. Finally, for each rule schema $\vec{X} \rightarrow Z$ in (11) we construct a context-free rule $Z \rightarrow \vec{X}$. Then, for each entry in the lexicon $w \rightarrow C$, we construct a context-free rule $C \rightarrow w$.

The constructed CFG has precisely the same rules as the CCG restricted to the categories in C except that the left and right sides have been reversed. Thus, by proposition 2, the CFG has exactly the same derivations as the CCG. □

Proposition 4. *Any CCG consisting of a subset of the schema classes (1-3), (6-8) and (10-11) along with restriction class (B) is strongly context-free.*

Proof. If a CCG is allowed to restrict the use of its rules to certain categories as in schema class (B), then when we construct the context-free rules by enumerating only those categories in the set C allowed by the restriction. □

Proposition 5. *Any CCG that includes restriction class (A) is strongly context-free.*

Proof. We construct a context-free grammar with exactly those rules in the finite set of instantiations of the CCG rule schemata along with context-free rules corresponding to the lexicon. This CFG generates exactly the same derivations as the CCG. □

We have thus proved that of a wide range of the rule schemata used to define CCGs are context-free.

2.3 Combinatory Categorial Grammars in Practice

CCGbank (Hockenmaier and Steedman, 2007) is a corpus of CCG derivations that was semi-automatically converted from the Wall Street Journal section of the Penn treebank. Figure 2 shows a categorization of the rules used in CCGbank according to the schema classes defined in the preceding section where a rule is placed into the least general class to which it belongs. In addition to having no generalized composition other than the reducing variant, it should also be noted that in all generalized composition rules, $X = Y$ implying that the reducing class of generalized composition is a very natural schema class for CCGbank.

If we assume that type-raising is restricted to those instances occurring in CCGbank[4], then a CCG consisting of schema classes (1-3), (6-7) and (10-11) can generate all the derivations in CCGbank. By proposition 3, such a CCG is strongly context-free. One could also observe that since CCGbank is finite, its grammar is not only a context-free grammar but can produce only a finite number of derivations. However, our statement is much stronger because this CCG can generate all of the derivations in CCGbank given only the lexicon, the finite set of unrestricted rules and the finite number of type-raising rules.

[4]Without such an assumption, parsing is intractable.

Schema Class	*Rules*	*Instances*
Application	519	902176
Composition	102	7189
Crossed Composition	64	14114
Reducing Generalized Crossed Composition	50	612
Generalized Composition	0	0
Generalized Crossed Composition	0	0
Substitution	3	4
Type-Raising	27	3996
Unrestricted Rules	642	335011
Total	1407	1263102

Figure 2: The rules of CCGbank by schema class.

The Clark and Curran CCG Parser (Clark and Curran, 2007) is a CCG parser which uses CCGbank as a training corpus. Despite the fact that there is a strongly context-free CCG which generates all of the derivations in CCGbank, it is still possible that the grammar learned by the Clark and Curran parser is not a context-free grammar. However, in addition to rule schemata (1-6) and (10-11) they also include restriction class (A) by restricting rules to only those found in the training data[5]. Thus, by proposition 5, the Clark and Curran parser is a context-free parser.

3 A Latent Variable CCG Parser

The context-freeness of a number of CCGs should not be considered evidence that there is no advantage to CCG as a grammar formalism. Unlike the context-free grammars extracted from the Penn treebank, these allow for the categorial semantics that accompanies any categorial parse and for a more elegant analysis of linguistic structures such as extraction and coordination. However, because we now know that the CCG defined by CCGbank is strongly context-free, we can use tools from the CFG parsing community to improve CCG parsing.

To illustrate this point, we train the Petrov parser (Petrov and Klein, 2007) on CCGbank. The Petrov parser uses latent variables to refine a coarse-grained grammar extracted from a training corpus to a grammar which makes much more fine-grained syntactic distinctions. For example, in Petrov's experiments on the Penn treebank, the syntactic category NP was refined to the more fine-grained NP^1 and NP^2 roughly corresponding to NPs in subject and object positions. Rather than requiring such distinctions to be made in the corpus, the Petrov parser hypothesizes these splits automatically.

The Petrov parser operates by performing a fixed number of iterations of splitting, merging and smoothing. The splitting process is done by performing Expectation-Maximization to determine a likely potential split for each syntactic category. Then, during the merging process some of the splits are undone to reduce grammar size and avoid overfitting according to the likelihood of the split against the training data.

The Petrov parser was chosen for our experiments because it refines the grammar in a mathematically principled way without altering the nature of the derivations that are output. This is important because the input to the semantic backend and the system that converts CCG derivations to dependencies requires CCG derivations as they appear in CCGbank.

3.1 Experiments

Our experiments use CCGbank as the corpus and we use sections 02-21 for training (39603 sentences), 00 for development (1913 sentences) and 23 for testing (2407 sentences).

CCGbank, in addition to the basic atoms S, N, NP and PP, also differentiates both the S and NP atoms with *features* allowing more subtle distinctions. For example, declarative sentences are $S[dcl]$, wh-questions are $S[wq]$ and sentence fragments are $S[frg]$ (Hockenmaier and Steedman, 2007). These features allow finer control of the use of combinatory rules in the resulting grammars. However, this fine-grained control is exactly what the Petrov parser does automatically. Therefore, we trained the Petrov parser twice, once on the original version of CCGbank (denoted "Petrov") and once on a version of CCGbank without these features (denoted "Petrov no feats"). Furthermore, we will evaluate the parsers obtained after 0, 4, 5 and 6 training iterations (denoted I-0, I-4, I-5 and I-6). When we evaluate on sets of sentences for which not all parsers return an analysis, we report the coverage (denoted "Cover").

We use the `evalb` package for PARSEVAL evaluation and a modified version of Clark and

[5]The Clark and Curran parser has an option, which is disabled by default, for not restricting the rules to those that appear in the training data. However, they find that this restriction is "detrimental to neither parser accuracy or coverage" (Clark and Curran, 2007).

Parser	*Accuracy %*	*No feats %*
C&C Normal Form	92.92	93.38
C&C Hybrid	93.06	93.52
Petrov I-5	**93.18**	93.73
Petrov no feats I-6	-	**93.74**

Figure 3: Supertagging accuracy on the sentences in section 00 that receive derivations from the four parsers shown.

Parser	*Accuracy %*	*No feats %*
C&C Hybrid	92.98	93.43
Petrov I-5	**93.10**	93.59
Petrov no feats I-6	-	**93.62**

Figure 4: Supertagging accuracy on the sentences in section 23 that receive derivations from the three parsers shown.

Curran's `evaluate` script for dependency evaluation. To determine statistical significance, we obtain p-values from Bikel's randomized parsing evaluation comparator[6], modified for use with tagging accuracy, F-score and dependency accuracy.

3.2 Supertag Evaluation

Before evaluating the parse trees as a whole, we evaluate the categories assigned to words. In the supertagging literature, POS tagging and supertagging are distinguished – POS tags are the traditional Penn treebank tags (e.g. NN, VBZ and DT) and supertags are CCG categories. However, because the Petrov parser trained on CCGbank has no notion of Penn treebank POS tags, we can only evaluate the accuracy of the supertags.

The results are shown in figures 3 and 4 where the "Accuracy" column shows accuracy of the supertags against the CCGbank categories and the "No feats" column shows accuracy when features are ignored. Despite the lack of POS tags in the Petrov parser, we can see that it performs slightly better than the Clark and Curran parser. The difference in accuracy is only statistically significant between Clark and Curran's Normal Form model ignoring features and the Petrov parser trained on CCGbank without features (p-value = 0.013).

3.3 Constituent Evaluation

In this section we evaluate the parsers using the traditional PARSEVAL measures which measure recall, precision and F-score on constituents in both labeled and unlabeled versions. In addition, we report a variant of the labeled PARSEVAL measures where we ignore the features on the categories. For reasons of brevity, we report the PARSEVAL measures for all sentences in sections 00 and 23, rather than for sentences of length is less than 40 or less than 100. The results are essentially identical for those two sets of sentences.

Figure 5 gives the PARSEVAL measures on section 00 for Clark and Curran's two best models and the Petrov parser trained on the original CCGbank and the version without features after various numbers of training iterations. Figure 7 gives the accuracies on section 23.

In the case of Clark and Curran's hybrid model, the poor accuracy relative to the Petrov parsers can be attributed to the fact that this model chooses derivations based on the associated dependencies at the expense of constituent accuracy (see section 3.4). In the case of Clark and Curran's normal form model, the large difference between labeled and unlabeled accuracy is primarily due to the mislabeling of a small number of features (specifically, NP[nb] and NP[num]). The labeled accuracies without features gives the results when features are disregarded.

Due to the similarity of the accuracies and the difference in the coverage between I-5 of the Petrov parser on CCGbank and I-6 of the Petrov parser on CCGbank without features, we reevaluate their results on only those sentences for which they both return derivations in figures 6 and 8. These results show that the features in CCGbank actually inhibit accuracy (to a statistically significant degree in the case of unlabeled accuracy on section 00) when used as training data for the Petrov parser.

Figure 9 gives a comparison between the Petrov parser trained on the Penn treebank and on CCGbank. These numbers should not be directly compared, but the similarity of the unlabeled measures indicates that the difference between the structure of the Penn treebank and CCGbank is not large.[7]

3.4 Dependency Evaluation

The constituent-based PARSEVAL measures are simple to calculate from the output of the Petrov parser but the relationship of the PARSEVAL

[6] http://www.cis.upenn.edu/ dbikel/software.html

[7] Because punctuation in CCG can have grammatical function, we include it in our accuracy calculations resulting in lower scores for the Petrov parser trained on the Penn treebank than those reported in Petrov and Klein (2007).

Parser	Labeled % R	Labeled % P	Labeled % F	Labeled no feats % R	Labeled no feats % P	Labeled no feats % F	Unlabeled % R	Unlabeled % P	Unlabeled % F	Cover
C&C Normal Form	71.14	70.76	70.95	80.66	80.24	80.45	86.16	85.71	85.94	98.95
C&C Hybrid	50.08	49.47	49.77	58.13	57.43	57.78	61.27	60.53	60.90	98.95
Petrov I-0	74.19	74.27	74.23	74.66	74.74	74.70	78.65	78.73	78.69	99.95
Petrov I-4	85.86	85.78	85.82	86.36	86.29	86.32	89.96	89.88	89.92	99.90
Petrov I-5	**86.30**	**86.16**	**86.23**	86.84	86.70	86.77	90.28	90.13	90.21	99.90
Petrov I-6	85.95	85.68	85.81	86.51	86.23	86.37	90.22	89.93	90.08	99.22
Petrov no feats I-0	-	-	-	72.16	72.59	72.37	76.52	76.97	76.74	99.95
Petrov no feats I-5	-	-	-	86.67	86.57	86.62	90.30	90.20	90.25	99.90
Petrov no feats I-6	-	-	-	**87.45**	**87.37**	**87.41**	**90.99**	**90.91**	**90.95**	99.84

Figure 5: Constituent accuracy on all sentences from section 00.

Parser	Labeled % R	Labeled % P	Labeled % F	Labeled no feats % R	Labeled no feats % P	Labeled no feats % F	Unlabeled % R	Unlabeled % P	Unlabeled % F
Petrov I-5	86.56	86.46	86.51	87.10	87.01	87.05	90.43	90.33	90.38
Petrov no feats I-6	-	-	-	**87.45**	**87.37**	**87.41**	**90.99**	**90.91**	**90.95**
p-value	-	-	-	0.089	0.090	0.088	0.006	0.008	0.007

Figure 6: Constituent accuracy on the sentences in section 00 that receive a derivation from both parsers.

Parser	Labeled % R	Labeled % P	Labeled % F	Labeled no feats % R	Labeled no feats % P	Labeled no feats % F	Unlabeled % R	Unlabeled % P	Unlabeled % F	Cover
C&C Normal Form	71.15	70.79	70.97	80.73	80.32	80.53	86.31	85.88	86.10	99.58
Petrov I-5	**86.94**	**86.80**	**86.87**	87.47	87.32	87.39	90.75	90.59	90.67	99.83
Petrov no feats I-6	-	-	-	**87.49**	**87.49**	**87.49**	**90.81**	**90.82**	**90.81**	99.96

Figure 7: Constituent accuracy on all sentences from section 23.

Parser	Labeled % R	Labeled % P	Labeled % F	Labeled no feats % R	Labeled no feats % P	Labeled no feats % F	Unlabeled % R	Unlabeled % P	Unlabeled % F
Petrov I-5	86.94	86.80	86.87	87.47	87.32	87.39	90.75	90.59	90.67
Petrov no feats I-6	-	-	-	**87.48**	**87.49**	**87.49**	**90.81**	**90.82**	**90.81**
p-value	-	-	-	0.463	0.215	0.327	0.364	0.122	0.222

Figure 8: Constituent accuracy on the sentences in section 23 that receive a derivation from both parsers.

Parser	Labeled % R	Labeled % P	Labeled % F	Unlabeled % R	Unlabeled % P	Unlabeled % F	Cover
Petrov on PTB I-6	89.65	89.97	89.81	90.80	91.13	90.96	100.00
Petrov on CCGbank I-5	86.94	86.80	86.87	90.75	90.59	90.67	99.83
Petrov on CCGbank no feats I-6	87.49	87.49	87.49	90.81	90.82	90.81	99.96

Figure 9: Constituent accuracy for the Petrov parser on the corpora on all sentences from Section 23.

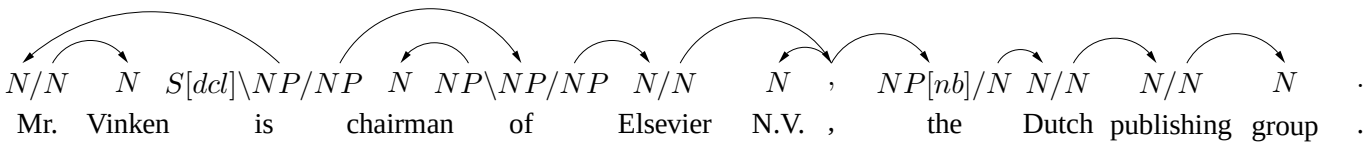


Figure 10: The argument-functor relations for the CCG derivation in figure 1.

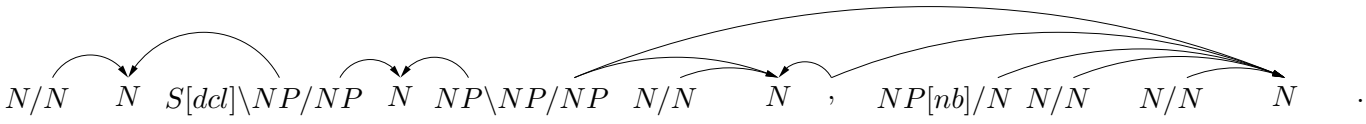


Figure 11: The set of dependencies obtained by reorienting the argument-functor edges in figure 10.

	Labeled %			*Unlabeled %*			
Parser	*R*	*P*	F	*R*	*P*	*F*	*Cover*
C&C Normal Form	84.39	85.28	84.83	90.93	91.89	91.41	98.95
C&C Hybrid	84.53	**86.20**	85.36	90.84	**92.63**	91.73	98.95
Petrov I-0	79.87	78.81	79.34	87.68	86.53	87.10	96.45
Petrov I-4	84.76	85.27	85.02	91.69	92.25	91.97	96.81
Petrov I-5	**85.30**	85.87	**85.58**	**92.00**	92.61	**92.31**	96.65
Petrov I-6	84.86	85.46	85.16	91.79	92.44	92.11	96.65

Figure 12: Dependency accuracy on CCGbank dependencies on all sentences from section 00.

	Labeled %			*Unlabeled %*		
Parser	*R*	*P*	F	*R*	*P*	*F*
C&C Hybrid	84.71	**86.35**	85.52	90.96	92.72	91.83
Petrov I-5	**85.50**	86.08	**85.79**	**92.12**	**92.75**	**92.44**
p-value	0.005	0.189	0.187	< 0.001	0.437	0.001

Figure 13: Dependency accuracy on the section 00 sentences that receive an analysis from both parsers.

	Labeled %			*Unlabeled %*		
Parser	*R*	*P*	F	*R*	*P*	*F*
C&C Hybrid	85.11	**86.46**	85.78	91.15	92.60	91.87
Petrov I-5	**85.73**	86.29	**86.01**	**92.04**	**92.64**	**92.34**
p-value	0.013	0.278	0.197	< 0.001	0.404	0.005

Figure 14: Dependency accuracy on the section 23 sentences that receive an analysis from both parsers.

Parser	*Training Time in CPU minutes*	*Parsing Time in CPU minutes*	*Training RAM in gigabytes*
Clark and Curran Normal Form Model	1152	2	28
Clark and Curran Hybrid Model	2672	4	37
Petrov on PTB I-0	1	5	2
Petrov on PTB I-5	180	20	8
Petrov on PTB I-6	660	21	16
Petrov on CCGbank I-0	1	5	2
Petrov on CCGbank I-4	103	70	8
Petrov on CCGbank I-5	410	600	14
Petrov on CCGbank I-6	2760	2880	24
Petrov on CCGbank no feats I-0	1	5	2
Petrov on CCGbank no feats I-5	360	240	7
Petrov on CCGbank no feats I-6	1980	390	13

Figure 15: Time and space usage when training on sections 02-21 and parsing on section 00.

scores to the quality of a parse is not entirely clear. For this reason, the word to word dependencies of categorial grammar parsers are often evaluated. This evaluation is aided by the fact that in addition to the CCG derivation for each sentence, CCGbank also includes a set of dependencies. Furthermore, extracting dependencies from a CCG derivation is well-established (Clark et al., 2002).

A CCG derivation can be converted into dependencies by, first, determining which arguments go with which functors as specified by the CCG derivation. This can be represented as in figure 10. Although this is not difficult, some care must be taken with respect to punctuation and the conjunction rules. Next, we reorient some of the edges according to information in the lexical categories. A language for specifying these instructions using variables and indices is given in Clark et al. (2002). This process is shown in figures 1, 10 and 11 with the directions of the dependencies reversed from Clark et al. (2002).

We used the CCG derivation to dependency converter `generate` included in the C&C tools package to convert the output of the Petrov parser to dependencies. Other than a CCG derivation, their system requires only the lexicon of edge reorientation instructions and methods for converting the unrestricted rules of CCGbank into the argument-functor relations. Important for the purpose of comparison, this system does not depend on their parser.

An unlabeled dependency is correct if the ordered pair of words is correct. A labeled dependency is correct if the ordered pair of words is correct, the head word has the correct category and the position of the category that is the source of that edge is correct. Figure 12 shows accuracies from the Petrov parser trained on CCGbank along with accuracies for the Clark and Curran parser. We only show accuracies for the Petrov parser trained on the original version of CCGbank because the dependency converter cannot currently generate dependencies for featureless derivations.

The relatively poor coverage of the Petrov parser is due to the failure of the dependency converter to output dependencies from valid CCG derivations. However, the coverage of the dependency converter is actually lower when run on the gold standard derivations indicating that this coverage problem is not indicative of inaccuracies in the Petrov parser. Due to the difference in coverage, we again evaluate the top two parsers on only those sentences that they both generate dependencies for and report those results in figures 13 and 14. The Petrov parser has better results by a statistically significant margin for both labeled and unlabeled recall and unlabeled F-score.

3.5 Time and Space Evaluation

As a final evaluation, we compare the resources that are required to both train and parse with the Petrov parser on the Penn Treebank, the Petrov parser on the original version of CCGbank, the Petrov parser on CCGbank without features and the Clark and Curran parser using the two models. All training and parsing was done on a 64-bit machine with 8 dual core 2.8 Ghz Opteron 8220 CPUs and 64GB of RAM. Our training times are much larger than those reported in Clark and Curran (2007) because we report the cumulative time spent on all CPUs rather than the maximum time spent on a CPU. Figure 15 shows the results.

As can be seen, the Clark and Curran parser has similar training times, although significantly greater RAM requirements than the Petrov parsers. In contrast, the Clark and Curran parser is significantly faster than the Petrov parsers, which we hypothesize to be attributed to the degree to which Clark and Curran have optimized their code, their use of `C++` as opposed to `Java` and their use of a supertagger to prune the lexicon.

4 Conclusion

We have provided a number of theoretical results proving that CCGbank contains no non-context-free structure and that the Clark and Curran parser is actually a context-free parser. Based on these results, we trained the Petrov parser on CCGbank and achieved state of the art results in terms of supertagging accuracy, PARSEVAL measures and dependency accuracy.

This demonstrates the following. First, the ability to extract semantic representations from CCG derivations is not dependent on the language class of a CCG. Second, using a dedicated supertagger, as opposed to simply using a general purpose tagger, is not necessary to accurately parse with CCG.

Acknowledgments

We would like to thank Stephen Clark, James Curran, Jackie C. K. Cheung and our three anonymous reviewers for their insightful comments.

References

J. Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

J. Bos, S. Clark, M. Steedman, J. R Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING*, volume 4, page 1240–1246.

S. Clark and J. R. Curran. 2007. Wide-Coverage efficient statistical parsing with CCG and Log-Linear models. *Computational Linguistics*, 33(4):493–552.

S. Clark, J. Hockenmaier, and M. Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, page 327–334.

J. Hockenmaier and M. Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

F. Hoyt and J. Baldridge. 2008. A logical basis for the d combinator and normal form in CCG. In *Proceedings of ACL-08: HLT*, page 326–334, Columbus, Ohio. Association for Computational Linguistics.

J. Lambek. 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, page 404–411.

M. Steedman. 2000. *The syntactic process*. MIT Press.

K. Vijay-Shanker and D. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

W. Zielonka. 1981. Axiomatizability of Ajdukiewicz-Lambek calculus by means of cancellation schemes. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 27:215–224.

Faster Parsing by Supertagger Adaptation

Jonathan K. Kummerfeld[a] **Jessica Roesner**[b] **Tim Dawborn**[a] **James Haggerty**[a]
James R. Curran[a*] **Stephen Clark**[c*]

School of Information Technologies[a] Department of Computer Science[b] Computer Laboratory[c]
University of Sydney University of Texas at Austin University of Cambridge
NSW 2006, Australia Austin, TX, USA Cambridge CB3 0FD, UK
james@it.usyd.edu.au[a*] stephen.clark@cl.cam.ac.uk[c*]

Abstract

We propose a novel self-training method for a parser which uses a lexicalised grammar and supertagger, focusing on increasing the speed of the parser rather than its accuracy. The idea is to train the supertagger on large amounts of parser output, so that the supertagger can learn to supply the supertags that the parser will eventually choose as part of the highest-scoring derivation. Since the supertagger supplies fewer supertags overall, the parsing speed is increased. We demonstrate the effectiveness of the method using a CCG supertagger and parser, obtaining significant speed increases on newspaper text with no loss in accuracy. We also show that the method can be used to adapt the CCG parser to new domains, obtaining accuracy and speed improvements for Wikipedia and biomedical text.

1 Introduction

In many NLP tasks and applications, e.g. distributional similarity (Curran, 2004) and question answering (Dumais et al., 2002), large volumes of text and detailed syntactic information are both critical for high performance. To avoid a trade-off between these two, we need to increase parsing speed, but without losing accuracy.

Parsing with lexicalised grammar formalisms, such as Lexicalised Tree Adjoining Grammar and Combinatory Categorial Grammar (CCG; Steedman, 2000), can be made more efficient using a *supertagger*. Bangalore and Joshi (1999) call supertagging *almost parsing* because of the significant reduction in ambiguity which occurs once the supertags have been assigned.

In this paper, we focus on the CCG parser and supertagger described in Clark and Curran (2007). Since the CCG lexical category set used by the supertagger is much larger than the Penn Treebank POS tag set, the accuracy of supertagging is much lower than POS tagging; hence the CCG supertagger assigns multiple supertags[1] to a word, when the local context does not provide enough information to decide on the correct supertag.

The supertagger feeds lexical categories to the parser, and the two interact, sometimes using multiple passes over a sentence. If a spanning analysis cannot be found by the parser, the number of lexical categories supplied by the supertagger is increased. The supertagger-parser interaction influences speed in two ways: first, the larger the lexical ambiguity, the more derivations the parser must consider; second, each further pass is as costly as parsing a whole extra sentence.

Our goal is to increase parsing speed without loss of accuracy. The technique we use is a form of *self-training*, in which the output of the parser is used to train the supertagger component. The existing literature on self-training reports mixed results. Clark et al. (2003) were unable to improve the accuracy of POS tagging using self-training. In contrast, McClosky et al. (2006a) report improved accuracy through self-training for a two-stage parser and re-ranker.

Here our goal is not to improve accuracy, only to maintain it, which we achieve through an *adaptive* supertagger. The adaptive supertagger produces lexical categories that the parser would have used in the final derivation when using the baseline model. However, it does so with much lower ambiguity levels, and potentially during an earlier pass, which means sentences are parsed faster. By increasing the ambiguity level of the adaptive models to match the baseline system, we can also slightly increase supertagging accuracy, which can lead to higher parsing accuracy.

[1] We use *supertag* and *lexical category* interchangeably.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 345–355,
Uppsala, Sweden, 11-16 July 2010.

Using the parser to generate training data also has the advantage that it is not a domain specific process. Previous work has shown that parsers typically perform poorly outside of their training domain (Gildea, 2001). Using a newspaper-trained parser, we constructed new training sets for Wikipedia and biomedical text. These were used to create new supertagging models adapted to the different domains.

The self-training method of adapting the supertagger to suit the parser increased parsing speed by more than 50% across all three domains, without loss of accuracy. Using an adapted supertagger with ambiguity levels tuned to match the baseline system, we were also able to increase F-score on labelled grammatical relations by 0.75%.

2 Background

Many statistical parsers use two stages: a tagging stage that labels each word with its grammatical role, and a parsing stage that uses the tags to form a parse tree. Lexicalised grammars typically contain a much smaller set of rules than phrase-structure grammars, relying on tags (supertags) that contain a more detailed description of each word's role in the sentence. This leads to much larger tag sets, and shifts a large proportion of the search for an optimal derivation to the tagging component of the parser.

Figure 1 gives two sentences and their CCG derivations, showing how some of the syntactic ambiguity is transferred to the supertagging component in a lexicalised grammar. Note that the lexical category assigned to with is different in each case, reflecting the fact that the prepositional phrase attaches differently. Either we need a tagging model that can resolve this ambiguity, or both lexical categories must be supplied to the parser which can then attempt to resolve the ambiguity by eventually selecting between them.

2.1 Supertagging

Supertaggers typically use standard linear-time tagging algorithms, and only consider words in the local context when assigning a supertag. The C&C supertagger is similar to the Ratnaparkhi (1996) tagger, using features based on words and POS tags in a five-word window surrounding the target word, and defining a local probability distribution over supertags for each word in the sentence, given the previous two supertags. The Viterbi algorithm

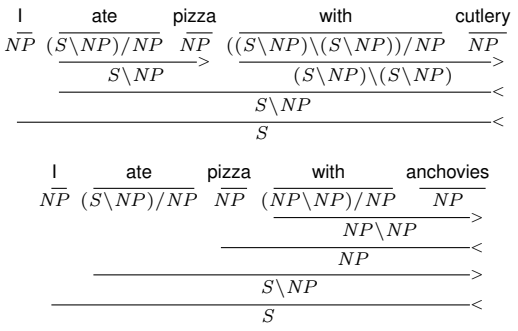


Figure 1: Two CCG derivations with PP ambiguity.

can be used to find the most probable supertag sequence. Alternatively the Forward-Backward algorithm can be used to efficiently sum over all sequences, giving a probability distribution over supertags for each word which is conditional only on the input sentence.

Supertaggers can be made accurate enough for wide coverage parsing using multi-tagging (Chen et al., 1999), in which more than one supertag can be assigned to a word; however, as more supertags are supplied by the supertagger, parsing efficiency decreases (Chen et al., 2002), demonstrating the influence of lexical ambiguity on parsing complexity (Sarkar et al., 2000).

Clark and Curran (2004) applied supertagging to CCG, using a flexible multi-tagging approach. The supertagger assigns to a word all lexical categories whose probabilities are within some factor, β, of the most probable category for that word. When the supertagger is integrated with the C&C parser, several progressively lower β values are considered. If a sentence is not parsed on one pass then the parser attempts to parse the sentence again with a lower β value, using a larger set of categories from the supertagger. Since most sentences are parsed at the first level (in which the average number of supertags assigned to each word is only slightly greater than one), this provides some of the speed benefit of single tagging, but without loss of coverage (Clark and Curran, 2004).

Supertagging has since been effectively applied to other formalisms, such as HPSG (Blunsom and Baldwin, 2006; Zhang et al., 2009), and as an information source for tasks such as Statistical Machine Translation (Hassan et al., 2007). The use of parser output for supertagger training has been explored for LTAG by Sarkar (2007). However, the focus of that work was on improving parser and supertagger accuracy rather than speed.

Previously	,	watch	imports	were	denied	such	duty-free	treatment
S/S	,	N/N	**N**	$(S[dcl]\backslash NP)/(S[pss]\backslash NP)$	$(S[pss]\backslash NP)/NP$	**NP/NP**	**N/N**	**N**
N		**N**		$(\mathbf{S[dcl]\backslash NP})/\mathbf{NP}$	$S[pss]\backslash NP$	$(N/N)/(N/N)$		
$S[adj]\backslash NP$				$(S[dcl]\backslash NP)/(S[adj]\backslash NP)$	$(S[pss]\backslash NP)/NP$	N/N		
					$(S[pt]\backslash NP)/NP$			
					$(\mathbf{S[dcl]\backslash NP})/\mathbf{NP}$			

Figure 2: An example sentence and the sets of categories assigned by the supertagger. The first category in each column is correct and the categories used by the parser are marked in bold. The correct category for watch is included here, for expository purposes, but in fact was not provided by the supertagger.

2.2 Semi-supervised training

Previous exploration of semi-supervised training in NLP has focused on improving accuracy, often for the case where only small amounts of manually labelled training data are available. One approach is *co-training*, in which two models with independent views of the data iteratively inform each other by labelling extra training data. Sarkar (2001) applied co-training to LTAG parsing, in which the supertagger and parser provide the two views. Steedman et al. (2003) extended the method to a variety of parser pairs.

Another method is to use a re-ranker (Collins and Koo, 2002) on the output of a system to generate new training data. Like co-training, this takes advantage of a different view of the data, but the two views are not independent as the re-ranker is limited to the set of options produced by the system. This method has been used effectively to improve parsing performance on newspaper text (McClosky et al., 2006a), as well as adapting a Penn Treebank parser to a new domain (McClosky et al., 2006b).

As well as using independent views of data to generate extra training data, multiple views can be used to provide constraints at test time. Hollingshead and Roark (2007) improved the accuracy of a parsing pipeline by using the output of later stages to constrain earlier stages.

The only work we are aware of that uses self-training to improve the *efficiency* of parsers is van Noord (2009), who adopts a similar idea to the one in this paper for improving the efficiency of a Dutch parser based on a manually constructed HPSG grammar.

3 Adaptive Supertagging

The purpose of the supertagger is to cut down the search space for the parser by reducing the set of categories that must be considered for each word. A perfect supertagger would assign the correct category to every word. CCG supertaggers are about 92% accurate when assigning a single lexical category to each word (Clark and Curran, 2004). This is not accurate enough for wide coverage parsing and so a multi-tagging approach is used instead. In the final derivation, the parser uses one category from each set, and it is important to note that having the correct category in the set does not guarantee that the parser will use it.

Figure 2 gives an example sentence and the sets of lexical categories supplied by the supertagger, for a particular value of β.[2] The usual target of the supertagging task is to produce the top row of categories in Figure 2, the correct categories. We propose a new task that instead aims for the categories the parser will use, which are marked in bold for this case. The purpose of this new task is to improve speed.

The reason speed will be improved is that we can construct models that will constrain the set of possible derivations more than the baseline model. We can construct these models because we can obtain much more of our target output, parser-annotated sentences, than we could for the gold-standard supertagging task.

The new target data will contain tagging errors, and so supertagging accuracy measured against the correct categories may decrease. If we obtained perfect accuracy on our new task then we would be removing all of the categories not chosen by the parser. However, parsing accuracy will not decrease since the parser will still receive the categories it would have used, and will therefore be able to form the same highest-scoring derivation (and hence will choose it).

To test this idea we parsed millions of sentences

[2]Two of the categories for such have been left out for reasons of space, and the correct category for watch has been included for expository reasons. The fact that the supertagger does not supply this category is the reason that the parser does not analyse the sentence correctly.

in three domains, producing new data annotated with the categories that the parser used with the baseline model. We constructed new supertagging models that are adapted to suit the parser by training on the combination of these sets and the standard training corpora. We applied standard evaluation metrics for speed and accuracy, and explored the source of the changes in parsing performance.

4 Data

In this work, we consider three domains: newswire, Wikipedia text and biomedical text.

4.1 Training and accuracy evaluation

We have used Sections 02-21 of CCGbank (Hockenmaier and Steedman, 2007), the CCG version of the Penn Treebank (Marcus et al., 1993), as training data for the newspaper domain. Sections 00 and 23 were used for development and test evaluation. A further 113,346,430 tokens (4,566,241 sentences) of raw data from the Wall Street Journal section of the North American News Corpus (Graff, 1995) were parsed to produce the training data for adaptation. This text was tokenised using the C&C tools tokeniser and parsed using our baseline models. For the smaller training sets, sentences from 1988 were used as they would be most similar in style to the evaluation corpus. In all experiments the sentences from 1989 were excluded to ensure no overlap occurred with CCGbank.

As Wikipedia text we have used 794,024,397 tokens (51,673,069 sentences) from Wikipedia articles. This text was processed in the same way as the NANC data to produce parser-annotated training data. For supertagger evaluation, one thousand sentences were manually annotated with CCG lexical categories and POS tags. For parser evaluation, three hundred of these sentences were manually annotated with DepBank grammatical relations (King et al., 2003) in the style of Briscoe and Carroll (2006). Both sets of annotations were produced by manually correcting the output of the baseline system. The annotation was performed by Stephen Clark and Laura Rimell.

For the biomedical domain we have used several different resources. As gold standard data for supertagger evaluation we have used supertagged GENIA data (Kim et al., 2003), annotated by Rimell and Clark (2008). For parsing evaluation, grammatical relations from the BioInfer corpus were used (Pyysalo et al., 2007), with the same post-processing process as Rimell and Clark (2009) to convert the C&C parser output to Stanford format grammatical relations (de Marneffe et al., 2006). For adaptive training we have used 1,900,618,859 tokens (76,739,723 sentences) from the MEDLINE abstracts tokenised by McIntosh and Curran (2008). These sentences were POS-tagged and parsed twice, once as for the newswire and Wikipedia data, and then again, using the bio-specific models developed by Rimell and Clark (2009). Statistics for the sentences in the training sets are given in Table 1.

Source	Sentence Length			Corpus %
	Range	Average	Variance	
News	0-4	3.26	0.64	1.2
	5-20	14.04	17.41	39.2
	21-40	28.76	29.27	49.4
	41-250	49.73	86.73	10.2
	All	24.83	152.15	100.0
Wiki	0-4	2.81	0.60	22.4
	5-20	11.64	21.56	48.9
	21-40	28.02	28.48	24.3
	41-250	49.69	77.70	4.5
	All	15.33	154.57	100.0
Bio	0-4	2.98	0.75	0.9
	5-20	14.54	15.14	41.3
	21-40	28.49	29.34	48.0
	41-250	49.17	68.34	9.8
	All	24.53	139.35	100.0

Table 1: Statistics for sentences in the supertagger training data. Sentences containing more than 250 tokens were not included in our data sets.

4.2 Speed evaluation data

For speed evaluation we held out three sets of sentences from each domain-specific corpus. Specifically, we used 30,000, 4,000 and 2,000 unique sentences of length 5-20, 21-40 and 41-250 tokens respectively. Speeds on these length controlled sets were combined to calculate an overall parsing speed for the text in each domain. Note that more than 20% of the Wikipedia sentences were less than five words in length and the overall distribution is skewed towards shorter sentences compared to the other corpora.

5 Evaluation

We used the hybrid parsing model described in Clark and Curran (2007), and the Viterbi decoder to find the highest-scoring derivation. The multi-pass supertagger-parser interaction was also used.

The test data was excluded from training data for the supertagger for all of the newswire and Wikipedia models. For the biomedical models ten-

fold cross validation was used. The accuracy of supertagging is measured by multi-tagging at the first β level and considering a word correct if the correct tag is amongst any of the assigned tags.

For the biomedical parser evaluation we have used the parsing model and grammatical relation conversion script from Rimell and Clark (2009).

Our timing measurements are calculated in two ways. Overall times were measured using the C&C parser's timers. Individual sentence measurements were made using the Intel timing registers, since standard methods are not accurate enough for the short time it takes to parse a single sentence.

To check whether changes were statistically significant we applied the test described by Chinchor (1995). This measures the probability that two sets of responses are drawn from the same distribution, where a score below 0.05 is considered significant.

Models were trained on an Intel Core2Duo 3GHz with 4GB of RAM. The evaluation was performed on a dual quad-core Intel Xeon 2.27GHz with 16GB of RAM.

5.1 Tagging ambiguity optimisation

The number of lexical categories assigned to a word by the CCG supertagger depends on the probabilities calculated for each category and the β level being used. Each lexical category with a probability within a factor of β of the most probable category is included. This means that the choice of β level determines the tagging ambiguity, and so has great influence on parsing speed, accuracy and coverage. Also, the tagging ambiguity produced by a β level will vary between models. A more confident model will have a more peaked distribution of category probabilities for a word, and therefore need a smaller β value to assign the same number of categories.

Additionally, the C&C parser uses multiple β levels. The first pass over a sentence is at a high β level, resulting in a low tagging ambiguity. If the categories assigned are too restrictive to enable a spanning analysis, the system makes another pass with a lower β level, resulting in a higher tagging ambiguity. A maximum of five passes are made, with the β levels varying from 0.075 to 0.001.

We have taken two approaches to choosing β levels. When the aim of an experiment is to improve speed, we use the system's default β levels. While this choice means a more confident model will assign fewer tags, this simply reflects the fact that the model is more confident. It should produce similar accuracy results, but with lower ambiguity, which will lead to higher speed.

For accuracy optimisation experiments we tune the β levels to produce the same average tagging ambiguity as the baseline model on Section 00 of CCGbank. Accuracy depends heavily on the number of categories supplied, so the new models are at an accuracy disadvantage if they propose fewer categories. By matching the ambiguity of the default model, we can increase accuracy at the cost of some of the speed improvements the new models obtain.

6 Results

We have performed four primary sets of experiments to explore the ability of an adaptive supertagger to improve parsing speed or accuracy. In the first two experiments, we explore performance on the newswire domain, which is the source of training data for the parsing model and the baseline supertagging model. In the second set of experiments, we train on a mixture of gold standard newswire data and parser-annotated data from the target domain.

In both cases we perform two experiments. The first aimed to improve speed, keeping the β levels the same. This should lead to an increase in speed as the extra training data means the models are more confident and so have lower ambiguity than the baseline model for a given β value. The second experiment aimed to improve accuracy, tuning the β levels as described in the previous section.

6.1 Newswire speed improvement

In our first experiment, we trained supertagger models using Generalised Iterative Scaling (GIS) (Darroch and Ratcliff, 1972), the limited memory BFGS method (BFGS) (Nocedal and Wright, 1999), the averaged perceptron (Collins, 2002), and the margin infused relaxed algorithm (MIRA) (Crammer and Singer, 2003). Note that these are all alternative methods for estimating the *local* log-linear probability distributions used by the Ratnaparkhi-style tagger. We do not use global tagging models as in Lafferty et al. (2001) or Collins (2002). The training data consisted of Sections 02–21 of CCGbank and progressively larger quantities of parser-annotated NANC data – from zero to four million extra sentences. The results of these tests are presented in Table 2.

	Ambiguity (%)				Tagging Accuracy (%)				F-score				Speed (sents / sec)			
Data	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m
Baseline	1.27				96.34				85.46				39.6			
BFGS	1.27	1.23	1.19	1.18	96.33	96.18	95.95	95.93	85.45	85.51	85.57	85.68	39.8	49.6	71.8	60.0
GIS	1.28	1.24	1.21	1.20	96.44	96.27	96.09	96.11	85.44	85.46	85.58	85.62	37.4	44.1	51.3	54.1
MIRA	1.30	1.24	1.17	1.13	96.44	96.14	95.56	95.18	85.44	85.40	85.38	85.42	34.1	44.8	60.2	**73.3**

Table 2: Speed improvements on newswire, using various amounts of parser-annotated NANC data.

		Sentences			Av. Time Change (ms)			Total Time Change (s)		
	Sentence length	5-20	21-40	41-250	5-20	21-40	41-250	5-20	21-40	41-250
Earlier pass	Lower tag amb.	1166	333	281	-7.54	-71.42	-183.23	-1.1	-29	-26
	Same tag amb.	248	38	8	-2.94	-27.08	-108.28	-0.095	-1.3	-0.44
	Higher tag amb.	530	33	14	-5.84	-32.25	-44.10	-0.40	-1.3	-0.31
Same pass	Lower tag amb.	19288	3120	1533	-1.13	-5.18	-38.05	-2.8	-20	-30
	Same tag amb.	7285	259	35	-0.29	0.94	24.57	-0.28	0.30	0.44
	Higher tag amb.	1133	101	24	-0.25	2.70	8.09	-0.037	0.34	0.099
Later pass	Lower tag amb.	334	114	104	0.90	7.60	-46.34	0.039	1.1	-2.5
	Same tag amb.	14	1	0	1.06	4.26	n/a	0.0019	0.0053	0.0
	Higher tag amb.	2	1	1	-0.13	26.43	308.03	-3.4e-05	0.033	0.16

Table 3: Breakdown of the source of changes in speed. The test sentences are divided into nine sets based on the change in parsing behaviour between the baseline model and a model trained using MIRA, Sections 02-21 of CCGbank and 4,000,000 NANC sentences.

Using the default β levels we found that the perceptron-trained models lost accuracy, disqualifying them from this test. The BFGS, GIS and MIRA models produced mixed results, but no statistically significant decrease in accuracy, and as the amount of parser-annotated data was increased, parsing speed increased by up to 85%.

To determine the source of the speed improvement we considered the times recorded by the timing registers. In Table 3, we have aggregated these measurements based on the change in the pass at which the sentence is parsed, and how the tagging ambiguity changes on that pass. For sentences parsed on two different passes the ambiguity comparison is at the earlier pass. The "Total Time Change" section of the table is the change in parsing time for sentences of that type when parsing ten thousand sentences from the corpus. This takes into consideration the actual distribution of sentence lengths in the corpus.

Several effects can be observed in these results. 72% of sentences are parsed on the same pass, but with lower tag ambiguity (5th row in Table 3). This provides 44% of the speed improvement. Three to six times as many sentences are parsed on an earlier pass than are parsed on a later pass. This means the sentences parsed later have very little effect on the overall speed. At the same time, the average gain for sentences parsed earlier is almost always larger than the average cost for sentences parsed later. These effects combine to produce a particularly large improvement for the sentences parsed at an earlier pass. In fact, despite making up only 7% of sentences in the set, those parsed earlier with lower ambiguity provide 50% of the speed improvement.

It is also interesting to note the changes for sentences parsed on the same pass, with the same ambiguity. We may expect these sentences to be parsed in approximately the same amount of time, and this is the case for the short set, but not for the two larger sets, where we see an increase in parsing time. This suggests that the categories being supplied are more productive, leading to a larger set of possible derivations.

6.2 Newswire accuracy optimised

Any decrease in tagging ambiguity will generally lead to a decrease in accuracy. The parser uses a more sophisticated algorithm with global knowledge of the sentence and so we would expect it to be better at choosing categories than the supertagger. Unlike the supertagger it will exclude categories that cannot be used in a derivation. In the previous section, we saw that training the supertagger on parser output allowed us to develop models that produced the same categories, despite lower tagging ambiguity. Since they were trained on the categories the parser was able to use in derivations, these models should also now be providing categories that are more likely to be useful.

This leads us to our second experiment, opti-

	Tagging Accuracy (%)				F-score				Speed (sents / sec)			
NANC sents	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m
Baseline	96.34				85.46				39.6			
BFGS	96.33	96.42	96.42	**96.66**	85.45	85.55	85.64	**85.98**	39.5	43.7	43.9	42.7
GIS	96.34	96.43	96.53	96.62	85.36	85.47	85.84	85.87	39.1	41.4	41.7	42.6
Perceptron	95.82	95.99	96.30	-	85.28	85.39	85.64	-	45.9	48.0	45.2	-
MIRA	96.23	96.29	96.46	96.63	85.47	85.45	85.55	85.84	37.7	41.4	41.4	42.9

Table 4: Accuracy optimisation on newswire, using various amounts of parser-annotated NANC data.

Train Corpus	Ambiguity			Tag. Acc.			F-score			Speed (sents / sec)		
	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio
Baseline	1.267	1.317	1.281	96.34	94.52	90.70	85.46	80.8	75.0	39.6	50.9	35.1
News	1.126	1.151	1.130	95.18	93.56	90.07	85.42	81.2	75.2	**73.3**	**83.9**	**60.3**
Wiki	1.147	1.154	1.129	95.06	93.52	90.03	84.70	81.4	75.5	62.4	**73.9**	58.7
Bio	1.134	1.146	1.114	94.66	93.15	89.88	84.23	80.7	75.9	66.2	**90.4**	**59.3**

Table 5: Cross-corpus speed improvement, models trained with MIRA and 4,000,000 sentences. The highlighted values are the top speed for each evaluation set and results that are statistically indistinguishable from it.

mising accuracy on newswire. We used the same models as in the previous experiment, but tuned the β levels as described in Section 5.1.

Comparing Tables 2 and 4 we can see the influence of β level choice, and therefore tagging ambiguity. When the default β values were used ambiguity dropped consistently as more parser-annotated data was used, and category accuracy dropped in the same way. Tuning the β levels to match ambiguity produces the opposite trend.

Interestingly, while the decrease in supertag accuracy in the previous experiment did not translate into a decrease in F-score, the increase in tag accuracy here does translate into an increase in F-score. This indicates that the supertagger is adapting to suit the parser. In the previous experiment, the supertagger was still providing the categories the parser would have used with the baseline supertagging model, but it provided fewer other categories. Since the parser is not a perfect supertagger these other categories may in fact have been incorrect, and so supertagger accuracy goes down, without changing parsing results. Here we have allowed the supertagger to assign extra categories, which will only increase its accuracy.

The increase in F-score has two sources. First, our supertagger is more accurate, and so the parser is more likely to receive category sets that can be combined into the correct derivation. Also, the supertagger has been trained on categories that the parser is able to use in derivations, which means they are more productive.

As Table 6 shows, this change translates into an improvement of up to 0.75% in F-score on Section 23 of CCGbank. All of the new models in the table make a statistically significant improvement over the baseline.

Model	Tag. Acc. (%)	F-score (%)	Speed (sents/sec)
Baseline	96.51	85.20	39.6
GIS, 4,000k NANC	96.83	**85.95**	42.6
BFGS, 4,000k NANC	96.91	85.90	42.7
MIRA, 4,000k NANC	96.84	85.79	42.9

Table 6: Evaluation of top models on Section 23 of CCGbank. All changes in F-score are statistically significant.

It is also interesting to note that the results in Tables 2, 4 and 6, are similar for all of the training algorithms. However, the training times differ considerably. For all four algorithms the training time is proportional to the amount of data, but the GIS and BFGS models trained on only CCGbank took 4,500 and 4,200 seconds to train, while the equivalent perceptron and MIRA models took 90 and 95 seconds to train.

6.3 Annotation method comparison

To determine whether these improvements were dependent on the annotations being produced by the parser we performed a set of tests with supertagger, rather than parser, annotated data. Three extra training sets were created by annotating newswire sentences with supertags using the baseline supertagging model. One set used the one-best tagger, and two were produced using the most probable tag for each word out of the set supplied by the multi-tagger, with variations in the β value and dictionary cutoff for the two sets.

Train Corpus	Ambiguity		Tag. Acc.			F-score			Speed (sents / sec)		
	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio
Baseline	1.317	1.281	96.34	94.52	90.70	85.46	80.8	75.0	39.6	50.9	35.1
News	1.331	1.322	**96.53**	**94.86**	**91.32**	**85.84**	80.1	75.2	41.8	32.6	31.4
Wiki	1.293	1.251	96.28	94.79	91.08	85.02	**81.7**	75.8	40.4	37.2	37.2
Bio	1.287	1.195	96.15	94.28	91.03	84.95	80.6	**76.1**	39.2	52.9	26.2

Table 7: Cross-corpus accuracy optimisation, models trained with GIS and 400,000 sentences.

Annotation method	Tag. Acc.	F-score
Baseline	96.34	85.46
Parser	96.46	85.55
One-best super	95.94	85.24
Multi-tagger a	95.91	84.98
Multi-tagger b	96.00	84.99

Table 8: Comparison of annotation methods for extra data. The multi-taggers used β values 0.075 and 0.001, and dictionary cutoffs 20 and 150, for taggers a and b respectively.

Corpus	Speed (sents / sec)		
Sent length	5-20	21-40	41-250
News	242	44.8	8.24
Wiki	224	42.0	6.10
Bio	268	41.5	6.48

Table 9: Cross-corpus speed for the baseline model on data sets balanced on sentence length.

As Table 8 shows, in all cases the use of supertagger-annotated data led to poorer performance than the baseline system, while the use of parser-annotated data led to an improvement in F-score. The parser has access to a range of information that the supertagger does not, producing a different view of the data that the supertagger can productively learn from.

6.4 Cross-domain speed improvement

When applying parsers out of domain they are typically slower and less accurate (Gildea, 2001). In this experiment, we attempt to increase speed on out-of-domain data. Note that for some of the results presented here it may appear that the C&C parser does not lose speed when out of domain, since the Wikipedia and biomedical corpora contain shorter sentences on average than the news corpus. However, by testing on balanced sets it is clear that speed does decrease, particularly for longer sentences, as shown in Table 9.

For our domain adaptation development experiments, we considered a collection of different models; here we only present results for the best set of models. For speed improvement these were MIRA models trained on 4,000,000 parser-annotated sentences from the target domain.

As Table 5 shows, this training method produces models adapted to the new domain. In particular, note that models trained on Wikipedia or the biomedical data produce lower F-scores[3] than the baseline on newswire. Meanwhile, on the target domain they are adapted to, these models achieve a higher F-score and parse sentences at least 45% faster than the baseline.

The changes in tagging ambiguity and accuracy also show that adaptation has occurred. In all cases, the new models have lower tagging ambiguity, and lower supertag accuracy. However, on the corpus of the extra data, the performance of the adapted models is comparable to the baseline model, which means the parser is probably still be receiving the same categories that it used from the sets provided by the baseline system.

6.5 Cross-domain accuracy optimised

The ambiguity tuning method used to improve accuracy on the newspaper domain can also be applied to the models trained on other domains. In Table 7, we have tested models trained using GIS and 400,000 sentences of parsed target-domain text, with β levels tuned to match ambiguity with the baseline.

As for the newspaper domain, we observe increased supertag accuracy and F-score. Also, in almost every case the new models perform worse than the baseline on domains other than the one they were trained on.

In some cases the models in Table 7 are less accurate than those in Table 5. This is because as well as optimising the β levels we have changed training methods. All of the training methods were tried, but only the method with the best results in newswire is included here, which for F-score when trained on 400,000 sentences was GIS.

The accuracy presented so far for the biomedi-

[3]Note that the F-scores for Wikipedia and biomedical text are reported to only three significant figures as only 300 and 500 sentences respectively were available for parser evaluation.

Train Corpus	F-score
Rimell and Clark (2009)	81.5
Baseline	80.7
CCGbank + Genia	81.5
+ Newswire	81.9
+ Wikipedia	82.2
+ Biomedical	81.7
+ R&C annotated Bio	82.3

Table 10: Performance comparison for models using extra gold standard biomedical data. Models were trained with GIS and 4,000,000 extra sentences, and are tested using a POS-tagger trained on biomedical data.

cal model is considerably lower than that reported by Rimell and Clark (2009). This is because no gold standard biomedical training data was used in our experiments. Table 10 shows the results of adding Rimell and Clark's gold standard biomedical supertag data and using their biomedical POS-tagger. The table also shows how accuracy can be further improved by adding our parser-annotated data from the biomedical domain as well as the additional gold standard data.

7 Conclusion

This work has demonstrated that an adapted supertagger can improve parsing speed and accuracy. The purpose of the supertagger is to reduce the search space for the parser. By training the supertagger on parser output, we allow the parser to reach the derivation it would have found, sooner. This approach also enables domain adaptation, improving speed and accuracy outside the original domain of the parser.

The perceptron-based algorithms used in this work are also able to function online, modifying the model weights after each sentence is parsed. This could be used to construct a system that continuously adapts to the domain it is parsing.

By training on parser-annotated NANC data we constructed models that were adapted to the newspaper-trained parser. The fastest model parsed sentences 1.85 times as fast and was as accurate as the baseline system. Adaptive training is also an effective method of improving performance on other domains. Models trained on parser-annotated Wikipedia text and MEDLINE text had improved performance on these target domains, in terms of both speed and accuracy. Optimising for speed or accuracy can be achieved by modifying the β levels used by the supertagger, which controls the lexical category ambiguity at each level used by the parser.

The result is an accurate and efficient wide-coverage CCG parser that can be easily adapted for NLP applications in new domains without manually annotating data.

Acknowledgements

We thank the reviewers for their helpful feedback. This work was supported by Australian Research Council Discovery grants DP0665973 and DP1097291, the Capital Markets Cooperative Research Centre, and a University of Sydney Merit Scholarship. Part of the work was completed at the Johns Hopkins University Summer Workshop and (partially) supported by National Science Foundation Grant Number IIS-0833652.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of the 21st International Conference on Computational Linguistics*, Sydney, Australia.

John Chen, Srinivas Bangalore, and Vijay K. Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 188–195, Bergen, Norway.

John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. 2002. Reranking an n-gram supertagger. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 259–268, Venice, Italy.

Nancy Chinchor. 1995. Statistical significance of MUC-6 results. In *Proceedings of the Sixth Message Understanding Conference*, pages 39–43, Columbia, MD, USA.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, James R. Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of the seventh Conference on Natural Language Learning*, pages 49–55, Edmonton, Canada.

Michael Collins and Terry Koo. 2002. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, PA, USA.

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

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

John N. Darroch and David Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–54, Genoa, Italy.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th International ACMSIGIR Conference on Research and Development*, Tampere, Finland.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, PA, USA.

David Graff. 1995. North American News Text Corpus. LDC95T21. Linguistic Data Consortium. Philadelphia, PA, USA.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 288–295, Prague, Czech Republic.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, pages 952–959, Prague, Czech Republic.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):180–182.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Brooklyn, NY, USA.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia.

Tara McIntosh and James R. Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Workshop*, Hobart, Australia.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer.

Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007. On the unification of syntactic annotations under the Stanford dependency scheme: a case study on bioinfer and GENIA. In *Proceedings of the ACL workshop on biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA, USA.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 475–484, Honolulu, HI, USA.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Anoop Sarkar, Fel Xia, and Aravind K. Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proceedings of the COLING Workshop on Efficiency in Large-scale Parsing Systems*, pages 37–42, Luxembourg.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8, Pittsburgh, PA, USA.

Anoop Sarkar. 2007. Combining supertagging and lexicalized tree-adjoining grammar parsing. In Srinivas Bangalore and Aravind Joshi, editors, *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach*. MIT Press, Boston, MA, USA.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Stephen Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 331–338, Budapest, Hungary.

Geertjan van Noord. 2009. Learning efficient parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 817–825. Association for Computational Linguistics.

Yao-zhong Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. HPSG supertagging: A sequence labeling view. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 210–213, Paris, France.

Using Smaller Constituents Rather Than Sentences in Active Learning for Japanese Dependency Parsing

Manabu Sassano
Yahoo Japan Corporation
Midtown Tower,
9-7-1 Akasaka, Minato-ku,
Tokyo 107-6211, Japan
msassano@yahoo-corp.jp

Sadao Kurohashi
Graduate School of Informatics,
Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto 606-8501, Japan
kuro@i.kyoto-u.ac.jp

Abstract

We investigate active learning methods for Japanese dependency parsing. We propose active learning methods of using partial dependency relations in a given sentence for parsing and evaluate their effectiveness empirically. Furthermore, we utilize syntactic constraints of Japanese to obtain more labeled examples from precious labeled ones that annotators give. Experimental results show that our proposed methods improve considerably the learning curve of Japanese dependency parsing. In order to achieve an accuracy of over 88.3%, one of our methods requires only 34.4% of labeled examples as compared to passive learning.

1 Introduction

Reducing annotation cost is very important because supervised learning approaches, which have been successful in natural language processing, require typically a large number of labeled examples. Preparing many labeled examples is time consuming and labor intensive.

One of most promising approaches to this issue is active learning. Recently much attention has been paid to it in the field of natural language processing. Various tasks have been targeted in the research on active learning. They include word sense disambiguation, e.g., (Zhu and Hovy, 2007), POS tagging (Ringger et al., 2007), named entity recognition (Laws and Schütze, 2008), word segmentation, e.g., (Sassano, 2002), and parsing, e.g., (Tang et al., 2002; Hwa, 2004).

It is the main purpose of this study to propose methods of improving active learning for parsing by using a smaller constituent than a sentence as a unit that is selected at each iteration of active learning. Typically in active learning for parsing a sentence has been considered to be a basic unit for selection. Small constituents such as chunks have not been used in sample selection for parsing. We use Japanese dependency parsing as a target task in this study since a simple and efficient algorithm of parsing is proposed and, to our knowledge, active learning for Japanese dependency parsing has never been studied.

The remainder of the paper is organized as follows. Section 2 describes the basic framework of active learning which is employed in this research. Section 3 describes the syntactic characteristics of Japanese and the parsing algorithm that we use. Section 4 briefly reviews previous work on active learning for parsing and discusses several research challenges. In Section 5 we describe our proposed methods and others of active learning for Japanese dependency parsing. Section 6 describes experimental evaluation and discussion. Finally, in Section 7 we conclude this paper and point out some future directions.

2 Active Learning

2.1 Pool-based Active Learning

Our base framework of active learning is based on the algorithm of (Lewis and Gale, 1994), which is called *pool-based active learning*. Following their sequential sampling algorithm, we show in Figure 1 the basic flow of pool-based active learning. Various methods for selecting informative examples can be combined with this framework.

2.2 Selection Algorithm for Large Margin Classifiers

One of the most accurate approaches to classification tasks is an approach with large margin classifiers. Suppose that we are given data points $\{x_i\}$ such that the associated label y_i will be either -1 or 1, and we have a hyperplane of some large margin classifier defined by $\{x : f(x) = 0\}$ where the

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 356–365,
Uppsala, Sweden, 11-16 July 2010.

1. Build an initial classifier from an initial labeled training set.
2. While resources for labeling examples are available
   (a) Apply the current classifier to each unlabeled example
   (b) Find the m examples which are most *informative* for the classifier
   (c) Have annotators label the m examples
   (d) Train a new classifier on all labeled examples

Figure 1: Flow of the pool-based active learning

	Lisa-ga	kare-ni	ano	pen-wo	age-ta.
	Lisa-subj	to him	that	pen-acc	give-past.
ID	0	1	2	3	4
Head	4	4	3	4	-

Figure 2: Sample sentence. An English translation is "Lisa gave that pen to him."

classification function is $G(x) = \text{sign}\{f(x)\}$. In pool-based active learning with large margin classifiers, selection of examples can be done as follows:

1. Compute $f(x_i)$ over all unlabeled examples x_i in the pool.
2. Sort x_i with $|f(x_i)|$ in ascending order.
3. Select top m examples.

This type of selection methods with SVMs is discussed in (Tong and Koller, 2000; Schohn and Cohn, 2000). They obtain excellent results on text classification. These selection methods are simple but very effective.

3 Japanese Parsing

3.1 Syntactic Units

A basic syntactic unit used in Japanese parsing is a *bunsetsu*, the concept of which was initially introduced by Hashimoto (1934). We assume that in Japanese we have a sequence of *bunsetsu*s before parsing a sentence. A *bunsetsu* contains one or more content words and zero or more function words.

A sample sentence in Japanese is shown in Figure 2. This sentence consists of five bunsetsus: *Lisa-ga, kare-ni, ano, pen-wo,* and *age-ta* where *ga, ni,* and *wo* are postpositions and *ta* is a verb ending for past tense.

3.2 Constraints of Japanese Dependency Analysis

Japanese is a head final language and in written Japanese we usually hypothesize the following:

- Each bunsetsu has only one head except the rightmost one.
- Dependency links between bunsetsus go from left to right.
- Dependencies do not cross one another.

We can see that these constraints are satisfied in the sample sentence in Figure 2. In this paper we also assume that the above constraints hold true when we discuss algorithms of Japanese parsing and active learning for it.

3.3 Algorithm of Japanese Dependency Parsing

We use Sassano's algorithm (Sassano, 2004) for Japanese dependency parsing. The reason for this is that it is very accurate and efficient[1]. Furthermore, it is easy to implement. His algorithm is one of the simplest form of shift-reduce parsers and runs in linear-time.[2] Since Japanese is a head final language and its dependencies are projective as described in Section 3.2, that simplification can be made.

The basic flow of Sassano's algorithm is shown in Figure 3, which is slightly simplified from the original by Sassano (2004). When we use this algorithm with a machine learning-based classifier, function Dep() in Figure 3 uses the classifier to decide whether two bunsetsus have a dependency relation. In order to prepare training examples for the trainable classifier used with his algorithm, we first have to convert a treebank to suitable labeled instances by using the algorithm in Figure 4. Note

[1] Iwatate et al. (2008) compare their proposed algorithm with various ones that include Sassano's, cascaded chunking (Kudo and Matsumoto, 2002), and one in (McDonald et al., 2005). Kudo and Matsumoto (2002) compare cascaded chunking with the CYK method (Kudo and Matsumoto, 2000). After considering these results, we have concluded so far that Sassano's is a reasonable choice for our purpose.

[2] Roughly speaking, Sassano's is considered to be a simplified version, which is modified for head final languages, of Nivre's (Nivre, 2003). Classifiers with Nivre's are required to handle multiclass prediction, while binary classifiers can work with Sassano's for Japanese.

Input: w_i: bunsetsus in a given sentence.
N: the number of bunsetsus.
Output: h_j: the head IDs of bunsetsus w_j.
Functions: Push(i, s): pushes i on the stack s.
Pop(s): pops a value off the stack s.
Dep(j, i, w): returns true when w_j should modify w_i. Otherwise returns false.

```
procedure Analyze(w, N, h)
var s: a stack for IDs of modifier bunsetsus
begin
  {−1 indicates no modifier candidate}
  Push(−1, s);
  Push(0, s);
  for i ← 1 to N − 1 do begin
    j ← Pop(s);
    while (j ≠ −1
        and ((i = N − 1) or Dep(j, i, w)) ) do
    begin
      h_j ← i;
      j ← Pop(s)
    end
    Push(j, s);
    Push(i, s)
  end
end
```

Figure 3: Algorithm of Japanese dependency parsing

that the algorithm in Figure 4 does not generate every pair of bunsetsus.[3]

4 Active Learning for Parsing

Most of the methods of active learning for parsing in previous work use selection of sentences that seem to contribute to the improvement of accuracy (Tang et al., 2002; Hwa, 2004; Baldridge and Osborne, 2004). Although Hwa suggests that sample selection for parsing would be improved by selecting finer grained constituents rather than sentences (Hwa, 2004), such methods have not been investigated so far.

Typical methods of selecting sentences are

[3]We show a sample set of generated examples for training the classifier of the parser in Figure 3. By using the algorithm in Figure 4, we can obtain labeled examples from the sample sentences in Figure 2: $\{0, 1,$ "O"$\}$, $\{1, 2,$ "O"$\}$, $\{2, 3,$ "D"$\}$, and $\{1, 3,$ "O"$\}$. Please see Section 5.2 for the notation used here. For example, an actual labeled instance generated from $\{2, 3,$ "D"$\}$ will be like "label=D, features={modifier-content-word=ano, ..., head-content-word=pen, ...}."

Input: h_i: the head IDs of bunsetsus w_i.
Function: Dep(j, i, w, h): returns true if $h_j = i$.
Otherwise returns false. Also prints a feature vector with a label according to h_j.

```
procedure Generate(w, N, h)
begin
  Push(−1, s);
  Push(0, s);
  for i ← 1 to N − 1 do begin
    j ← Pop(s);
    while (j ≠ −1
      and ((i = N − 1) or Dep(j, i, w, h)) ) do
    begin
      j ← Pop(s)
    end
    Push(j, s);
    Push(i, s)
  end
end
```

Figure 4: Algorithm of generating training examples

based on some entropy-based measure of a given sentence (e.g., (Tang et al., 2002)). We cannot use this kind of measures when we want to select other smaller constituents than sentences. Other bigger problem is an algorithm of parsing itself. If we sample smaller units rather than sentences, we have partially annotated sentences and have to use a parsing algorithm that can be trained from incompletely annotated sentences. Therefore, it is difficult to use some of probabilistic models for parsing. [4]

5 Active Learning for Japanese Dependency Parsing

In this section we describe sample selection methods which we investigated.

5.1 Sentence-wise Sample Selection

Passive Selection (Passive) This method is to select sequentially sentences that appear in the training corpus. Since it gets harder for the readers to reproduce the same experimental setting, we

[4]We did not employ *query-by-committee* (QBC) (Seung et al., 1992), which is another important general framework of active learning, since the selection strategy with large margin classifiers (Section 2.2) is much simpler and seems more practical for active learning in Japanese dependency parsing with smaller constituents.

avoid to use random sampling in this paper.

Minimum Margin Selection (Min) This method is to select sentences that contain bunsetsu pairs which have smaller margin values of outputs of the classifier used in parsing. The procedure of selection of MIN are summarized as follows. Assume that we have sentences s_i in the pool of unlabeled sentences.

1. Parse s_i in the pool with the current model.
2. Sort s_i with $\min |f(x_k)|$ where x_k are bunsetsu pairs in the sentence s_i. Note that x_k are not all possible bunsetsu pairs in s_i and they are limited to bunsetsu pairs checked in the process of parsing s_i.
3. Select top m sentences.

Averaged Margin Selection (Avg) This method is to select sentences that have smaller values of averaged margin values of outputs of the classifier in a give sentences over the number of decisions which are carried out in parsing. The difference between AVG and MIN is that for AVG we use $\sum |f(x_k)|/l$ where l is the number of calling Dep() in Figure 3 for the sentence s_i instead of $\min |f(x_k)|$ for MIN.

5.2 Chunk-wise Sample Selection

In chunk-wise sample selection, we select bunsetsu pairs rather than sentences. Bunsetsu pairs are selected from different sentences in a pool. This means that structures of sentences in the pool are partially annotated.

Note that we do not use every bunsetsu pair in a sentence. When we use Sassano's algorithm, we have to generate training examples for the classifier by using the algorithm in Figure 4. In other words, we should not sample bunsetsu pairs independently from a given sentence.

Therefore, we select bunsetsu pairs that have smaller margin values of outputs given by the classifier during the parsing process. All the sentences in the pool are processed by the current parser. We cannot simply split the sentences in the pool into labeled and unlabeled ones because we do not select every bunsetsu pair in a given sentence.

Naive Selection (Naive) This method is to select bunsetsu pairs that have smaller margin values of outputs of the classifier. Then it is assumed that annotators would label either "D" for the two bunsetsu having a dependency relation or "O", which represents the two does not.

Modified Simple Selection (ModSimple) Although NAIVE seems to work well, it did not (discussed later). MODSIMPLE is to select bunsetsu pairs that have smaller margin values of outputs of the classifier, which is the same as in NAIVE. The difference between MODSIMPLE and NAIVE is the way annotators label examples. Assume that we have an annotator and the learner selects some bunsetsu pair of the j-th bunsetsu and the i-th bunsetsu such that $j < i$. The annotator is then asked what the head of the j-th bunsetsu is. We define here the head bunsetsu is the k-th one.

We differently generate labeled examples from the information annotators give according to the relation among bunsetsus j, i, and k.

Below we use the notation $\{s, t, \text{"D"}\}$ to denote that the s-th bunsetsu modifies the t-th one. The use of "O" instead of "D" indicates that the s-th does not modify the t-th. That is generating $\{s, t, \text{"D"}\}$ means outputting an example with the label "D".

Case 1 if $j < i < k$, then generate $\{j, i, \text{"O"}\}$ and $\{j, k, \text{"D"}\}$.

Case 2 if $j < i = k$, then generate $\{j, k, \text{"D"}\}$.

Case 3 if $j < k < i$, then generate $\{j, k, \text{"D"}\}$. Note that we do not generate $\{j, i, \text{"O"}\}$ in this case because in Sassano's algorithm we do not need such labeled examples if j depends on k such that $k < i$.

Syntactically Extended Selection (Syn) This selection method is one based on MODSIMPLE and extended to generate more labeled examples for the classifier. You may notice that more labeled examples for the classifier can be generated from a single label which the annotator gives. Syntactic constraints of the Japanese language allow us to extend labeled examples.

For example, suppose that we have four bunsetsus A, B, C, and D in this order. If A depends on C, i.e., the head of A is C, then it is automatically derived that B also should depend on C because the Japanese language has the no-crossing constraint for dependencies (Section 3.2). By utilizing this property we can obtain more labeled examples from a single labeled one annotators give. In the example above, we obtain $\{A, B, \text{"O"}\}$ and $\{B, C, \text{"D"}\}$ from $\{A, C, \text{"D"}\}$.

Although we can employ various extensions to MODSIMPLE, we use a rather simple extension in this research.

Case 1 if $(j < i < k)$, then generate

- $\{j, i, \text{“O”}\}$,
- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

Case 2 if $(j < i = k)$, then generate

- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

Case 3 if $(j < k < i)$, then generate

- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

In SYN as well as MODSIMPLE, we generate examples with ”O” only for bunsetsu pairs that occur to the left of the correct head (i.e., case 1).

6 Experimental Evaluation and Discussion

6.1 Corpus

In our experiments we used the Kyoto University Corpus Version 2 (Kurohashi and Nagao, 1998). Initial seed sentences and a pool of unlabeled sentences for training are taken from the articles on January 1st through 8th (7,958 sentences) and the test data is a set of sentences in the articles on January 9th (1,246 sentences). The articles on January 10th were used for development. The split of these articles for training/test/development is the same as in (Uchimoto et al., 1999).

6.2 Averaged Perceptron

We used the averaged perceptron (AP) (Freund and Schapire, 1999) with polynomial kernels. We set the degree of the kernels to 3 since cubic kernels with SVM have proved effective for Japanese dependency parsing (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2002). We found the best value of the epoch T of the averaged perceptron by using the development set. We fixed $T = 12$ through all experiments for simplicity.

6.3 Features

There are features that have been commonly used for Japanese dependency parsing among related papers, e.g., (Kudo and Matsumoto, 2002; Sassano, 2004; Iwatate et al., 2008). We also used the same features here. They are divided into three groups: modifier *bunsetsu* features, head *bunsetsu* features, and gap features. A summary of the features is described in Table 1.

6.4 Implementation

We implemented a parser and a tool for the averaged perceptron in C++ and used them for experiments. We wrote the main program of active learning and some additional scripts in Perl and sh.

6.5 Settings of Active Learning

For initial seed sentences, first 500 sentences are taken from the articles on January 1st. In experiments about sentence wise selection, 500 sentences are selected at each iteration of active learning and labeled[5] and added into the training data. In experiments about chunk wise selection 4000 pairs of bunsetsus, which are roughly equal to the averaged number of bunsetsus in 500 sentences, are selected at each iteration of active learning.

6.6 Dependency Accuracy

We use dependency accuracy as a performance measure of a parser. The dependency accuracy is the percentage of correct dependencies. This measure is commonly used for the Kyoto University Corpus.

6.7 Results and Discussion

Learning Curves First we compare methods for sentence wise selection. Figure 5 shows that MIN is the best among them, while AVG is not good and similar to PASSIVE. It is observed that active learning with large margin classifiers also works well for Sassano’s algorithm of Japanese dependency parsing.

Next we compare chunk-wise selection with sentence-wise one. The comparison is shown in Figure 6. Note that we must carefully consider how to count labeled examples. In sentence wise selection we obviously count the number of sentences. However, it is impossible to count such number when we label bunsetsus pairs.

Therefore, we use the number of bunsetsus that have an annotated head. Although we know this may not be a completely fair comparison, we believe our choice in this experiment is reasonable

[5] In our experiments human annotators do not give labels. Instead, labels are given virtually from correct ones that the Kyoto University Corpus has.

Bunsetsu features for modifiers and heads	rightmost content word, rightmost function word, punctuation, parentheses, location (BOS or EOS)
Gap features	distance (1, 2–5, or 6 $\leq$), particles, parentheses, punctuation

Table 1: Features for deciding a dependency relation between two bunsetsus. Morphological features for each word (morpheme) are major part-of-speech (POS), minor POS, conjugation type, conjugation form, and surface form.

for assessing the effect of reduction by chunk-wise selection.

In Figure 6 NAIVE has a better learning curve compared to MIN at the early stage of learning. However, the curve of NAIVE declines at the later stage and gets worse than PASSIVE and MIN.

Why does this phenomenon occur? It is because each bunsetsu pair is not independent and pairs in the same sentence are related to each other. They satisfy the constraints discussed in Section 3.2. Furthermore, the algorithm we use, i.e., Sassano's, assumes these constraints and has the specific order for processing bunsetsu pairs as we see in Figure 3. Let us consider the meaning of $\{j, i, \text{“O”}\}$ if the head of the j-th bunsetsu is the k-th one such that $j < k < i$. In the context of the algorithm in Figure 3, $\{j, i, \text{“O”}\}$ actually means that the j-th bunsetsu modifies th l-th one such that $i < l$. That is “O” does not simply mean that two bunsetsus does not have a dependency relation. Therefore, we should not generate $\{j, i, \text{“O”}\}$ in the case of $j < k < i$. Such labeled instances are not needed and the algorithm in Figure 4 does not generate them even if a fully annotated sentence is given. Based on the analysis above, we modified NAIVE and defined MODSIMPLE, where unnecessary labeled examples are not generated.

Now let us compare NAIVE with MODSIMPLE (Figure 7). MODSIMPLE is almost always better than PASSIVE and does not cause a significant deterioration of accuracy unlike NAIVE.[6]

Comparison of MODSIMPLE and SYN is shown in Figure 8. Both exhibit a similar curve. Figure 9 shows the same comparison in terms of required queries to human annotators. It shows that SYN is better than MODSIMPLE especially at the earlier stage of active learning.

Reduction of Annotations Next we examined the number of labeled bunsetsus to be required in

[6]We have to carefully see the curves of NAIVE and MODSIMPLE. In Figure 7 at the early stage NAIVE is slightly better than MODSIMPLE, while in Figure 9 NAIVE does not outperform MODSIMPLE. This is due to the difference of the way of accessing annotation efforts.

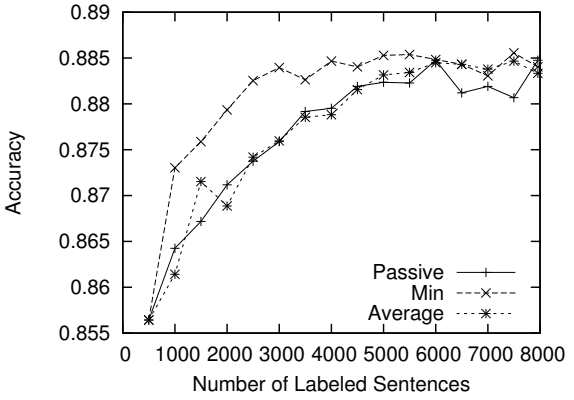


Figure 5: Learning curves of methods for sentence wise selection

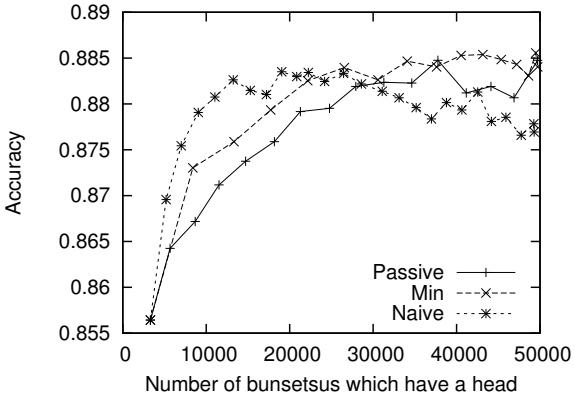


Figure 6: Learning curves of MIN (sentence-wise) and NAIVE (chunk-wise).

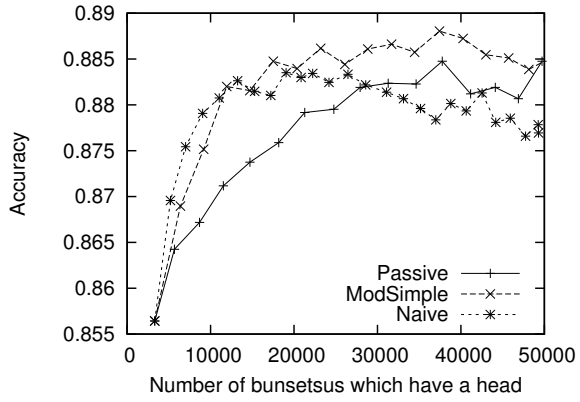


Figure 7: Learning curves of NAIVE, MODSIMPLE and PASSIVE in terms of the number of bunsetsus that have a head.

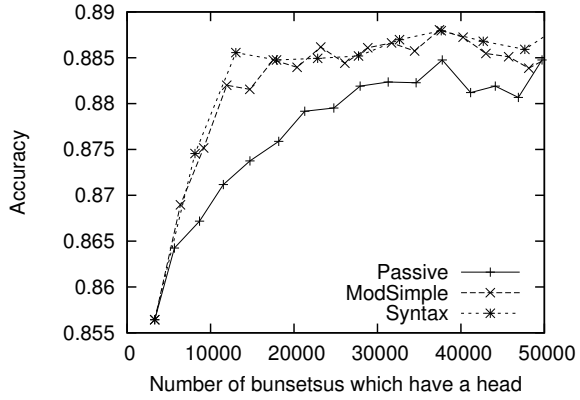


Figure 8: Learning curves of MODSIMPLE and SYN in terms of the number of bunsetsus which have a head.

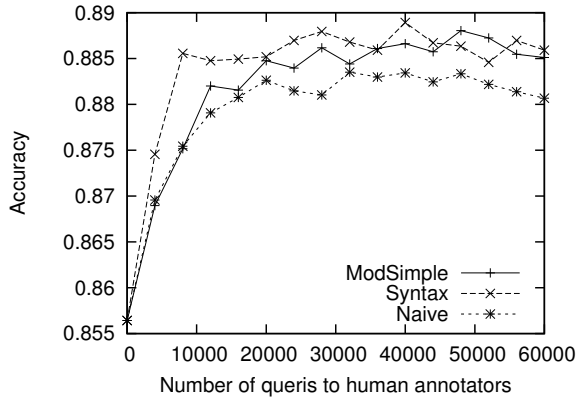


Figure 9: Comparison of MODSIMPLE and SYN in terms of the number of queries to human annotators

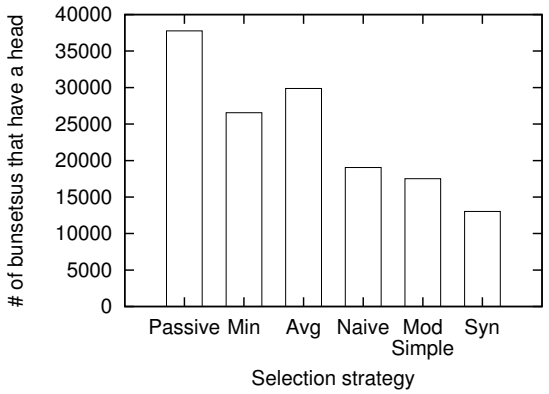


Figure 10: Number of labeled bunsetsus to be required to achieve an accuracy of over 88.3%.

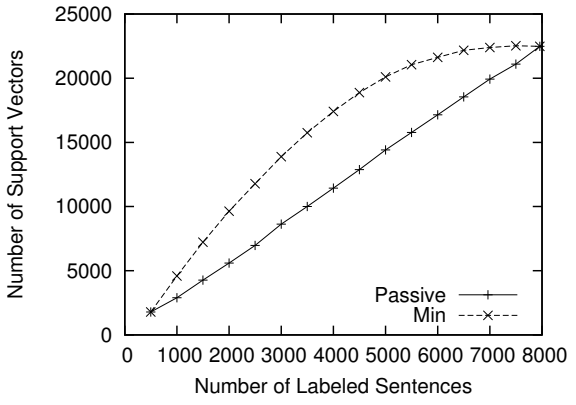


Figure 11: Changes of number of support vectors in sentence-wise active learning

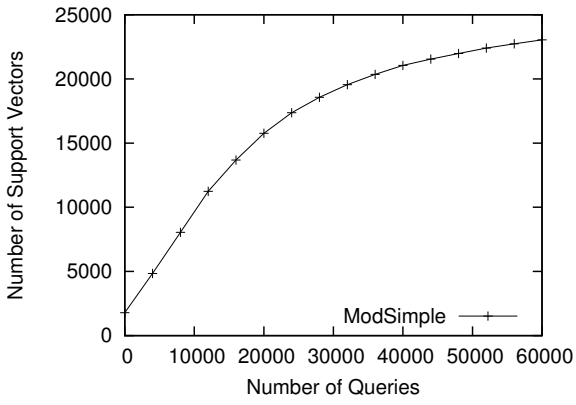


Figure 12: Changes of number of support vectors in chunk-wise active learning (MODSIMPLE)

order to achieve a certain level of accuracy. Figure 10 shows that the number of labeled bunsetsus to achieve an accuracy of over 88.3% depending on the active learning methods discussed in this research.

PASSIVE needs 37766 labeled bunsetsus which have a head to achieve an accuracy of 88.48%, while SYN needs 13021 labeled bunsetsus to achieve an accuracy of 88.56%. SYN requires only 34.4% of the labeled bunsetsu pairs that PASSIVE requires.

Stopping Criteria It is known that increment rate of the number of support vectors in SVM indicates saturation of accuracy improvement during iterations of active learning (Schohn and Cohn, 2000). It is interesting to examine whether the observation for SVM is also useful for support vectors[7] of the averaged perceptron. We plotted changes of the number of support vectors in the cases of both PASSIVE and MIN in Figure 11 and changes of the number of support vectors in the case of MODSIMPLE in Figure 12. We observed that the increment rate of support vectors mildly gets smaller. However, it is not so clear as in the case of text classification in (Schohn and Cohn, 2000).

Issues on Accessing the Total Cost of Annotation In this paper, we assume that each annotation cost for dependency relations is constant. It is however not true in an actual annotation work.[8] In addition, we have to note that it may be easier to annotate a whole sentence than some bunsetsu pairs in a sentence[9]. In a real annotation task, it will be better to show a whole sentence to annotators even when annotating some part of the sentence.

Nevertheless, it is noteworthy that our research shows the minimum number of annotations in preparing training examples for Japanese dependency parsing. The methods we have proposed must be helpful when checking repeatedly annotations that are important and might be wrong or difficult to label while building an annotated corpus. They also will be useful for domain adaptation of a dependency parser.[10]

Applicability to Other Languages and Other Parsing Algorithms We discuss here whether or not the proposed methods and the experiments are useful for other languages and other parsing algorithms. First we take languages similar to Japanese in terms of syntax, i.e., Korean and Mongolian. These two languages are basically head-final languages and have similar constraints in Section 3.2. Although no one has reported application of (Sassano, 2004) to the languages so far, we believe that similar parsing algorithms will be applicable to them and the discussion in this study would be useful.

On the other hand, the algorithm of (Sassano, 2004) cannot be applied to head-initial languages such as English. If target languages are assumed to be projective, the algorithm of (Nivre, 2003) can be used. It is highly likely that we will invent the effective use of finer-grained constituents, e.g., head-modifier pairs, rather than sentences in active learning for Nivre's algorithm with large margin classifiers since Sassano's seems to be a simplified version of Nivre's and they have several properties in common. However, syntactic constraints in European languages like English may be less helpful than those in Japanese because their dependency links do not have a single direction.

Even though the use of syntactic constraints is limited, smaller constituents will still be useful for other parsing algorithms that use some deterministic methods with machine learning-based classifiers. There are many algorithms that have such a framework, which include (Yamada and Matsumoto, 2003) for English and (Kudo and Matsumoto, 2002; Iwatate et al., 2008) for Japanese. Therefore, effective use of smaller constituents in active learning would not be limited to the specific algorithm.

7 Conclusion

We have investigated that active learning methods for Japanese dependency parsing. It is observed that active learning of parsing with the averaged perceptron, which is one of the large margin classifiers, works also well for Japanese dependency analysis.

[7] Following (Freund and Schapire, 1999), we use the term "support vectors" for AP as well as SVM. "Support vectors" of AP means vectors which are selected in the training phase and contribute to the prediction.

[8] Thus it is very important to construct models for estimating the actual annotation cost as Haertel et al. (2008) do.

[9] Hwa (2004) discusses similar aspects of researches on active learning.

[10] Ohtake (2006) examines heuristic methods of selecting sentences.

In addition, as far as we know, we are the first to propose the active learning methods of using partial dependency relations in a given sentence for parsing and we have evaluated the effectiveness of our methods. Furthermore, we have tried to obtain more labeled examples from precious labeled ones that annotators give by utilizing syntactic constraints of the Japanese language. It is noteworthy that linguistic constraints have been shown useful for reducing annotations in active learning for NLP.

Experimental results show that our proposed methods have improved considerably the learning curve of Japanese dependency parsing.

We are currently building a new annotated corpus with an annotation tool. We have a plan to incorporate our proposed methods to the annotation tool. We will use it to accelerate building of the large annotated corpus to improved our Japanese parser.

It would be interesting to explore the use of partially labeled constituents in a sentence in another language, e.g., English, for active learning.

Acknowledgements

We would like to thank the anonymous reviewers and Tomohide Shibata for their valuable comments.

References

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proc. of EMNLP 2004*, pages 9–16.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. 2008. Assessing the costs of sampling methods in active learning for annotation. In *Proc. of ACL-08: HLT, short papers (Companion Volume)*, pages 65–68.

Shinkichi Hashimoto. 1934. *Essentials of Japanese Grammar (Kokugoho Yousetsu) (in Japanese)*.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. 2008. Japanese dependency parsing using a tournament model. In *Proc. of COLING 2008*, pages 361–368.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of EMNLP/VLC 2000*, pages 18–25.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL-2002*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of LREC-1998*, pages 719–724.

Florian Laws and Hinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proc. of COLING 2008*, pages 465–472.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proc. of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL-2005*, pages 523–530.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-03*, pages 149–160.

Kiyonori Ohtake. 2006. Analysis of selective strategies to build a dependency-analyzed corpus. In *Proc. of COLING/ACL 2006 Main Conf. Poster Sessions*, pages 635–642.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proc. of the Linguistic Annotation Workshop*, pages 101–108.

Manabu Sassano. 2002. An empirical study of active learning with support vector machines for Japanese word segmentation. In *Proc. of ACL-2002*, pages 505–512.

Manabu Sassano. 2004. Linear-time dependency analysis for Japanese. In *Proc. of COLING 2004*, pages 8–14.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proc. of ICML-2000*, pages 839–846.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proc. of COLT '92*, pages 287–294.

Min Tang, Xaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proc. of ACL-2002*, pages 120–127.

Simon Tong and Daphne Koller. 2000. Support vector machine active learning with applications to text classification. In *Proc. of ICML-2000*, pages 999–1006.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proc. of EACL-99*, pages 196–203.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT 2003*, pages 195–206.

Jingbo Zhu and Eduard Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proc. of EMNLP-CoNLL 2007*, pages 783–790.

Conditional Random Fields for Word Hyphenation

Nikolaos Trogkanis
Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0404
tronikos@gmail.com

Charles Elkan
Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0404
elkan@cs.ucsd.edu

Abstract

Finding allowable places in words to insert hyphens is an important practical problem. The algorithm that is used most often nowadays has remained essentially unchanged for 25 years. This method is the TEX hyphenation algorithm of Knuth and Liang. We present here a hyphenation method that is clearly more accurate. The new method is an application of conditional random fields. We create new training sets for English and Dutch from the CELEX European lexical resource, and achieve error rates for English of less than 0.1% for correctly allowed hyphens, and less than 0.01% for Dutch. Experiments show that both the Knuth/Liang method and a leading current commercial alternative have error rates several times higher for both languages.

1 Introduction

The task that we investigate is learning to split words into parts that are conventionally agreed to be individual written units. In many languages, it is acceptable to separate these units with hyphens, but it is not acceptable to split words arbitrarily. Another way of stating the task is that we want to learn to predict for each letter in a word whether or not it is permissible for the letter to be followed by a hyphen. This means that we tag each letter with either 1, for hyphen allowed following this letter, or 0, for hyphen not allowed after this letter.

The hyphenation task is also called orthographic syllabification (Bartlett et al., 2008). It is an important issue in real-world text processing, as described further in Section 2 below. It is also useful as a preprocessing step to improve letter-to-phoneme conversion, and more generally for text-to-speech conversion. In the well-known NETtalk system, for example, syllable boundaries are an input to the neural network in addition to letter identities (Sejnowski and Rosenberg, 1988). Of course, orthographic syllabification is not a fundamental scientific problem in linguistics. Nevertheless, it is a difficult engineering task that is worth studying for both practical and intellectual reasons.

The goal in performing hyphenation is to predict a sequence of 0/1 values as a function of a sequence of input characters. This sequential prediction task is significantly different from a standard (non-sequential) supervised learning task. There are at least three important differences that make sequence prediction difficult. First, the set of all possible sequences of labels is an exponentially large set of possible outputs. Second, different inputs have different lengths, so it is not obvious how to represent every input by a vector of the same fixed length, as is almost universal in supervised learning. Third and most important, too much information is lost if we learn a traditional classifier that makes a prediction for each letter separately. Even if the traditional classifier is a function of the whole input sequence, this remains true. In order to achieve high accuracy, correlations between neighboring predicted labels must be taken into account.

Learning to predict a sequence of output labels, given a sequence of input data items, is an instance of a structured learning problem. In general, structured learning means learning to predict outputs that have internal structure. This structure can be modeled; to achieve high predictive accuracy, when there are dependencies between parts of an output, it must be modeled. Research on structured learning has been highly successful, with sequence classification as its most important and successful subfield, and with conditional random fields (CRFs) as the most influential approach to learning sequence classifiers. In the present paper,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 366–374,
Uppsala, Sweden, 11-16 July 2010.

we show that CRFs can achieve extremely good performance on the hyphenation task.

2 History of automated hyphenation

The earliest software for automatic hyphenation was implemented for RCA 301 computers, and used by the *Palm Beach Post-Tribune* and *Los Angeles Times* newspapers in 1962. These were two different systems. The Florida system had a dictionary of 30,000 words; words not in the dictionary were hyphenated after the third, fifth, or seventh letter, because the authors observed that this was correct for many words. The California system (Friedlander, 1968) used a collection of rules based on the rules stated in a version of Webster's dictionary. The earliest hyphenation software for a language other than English may have been a rule-based program for Finnish first used in 1964 (Jarvi, 2009).

The first formal description of an algorithm for hyphenation was in a patent application submitted in 1964 (Damerau, 1964). Other early publications include (Ocker, 1971; Huyser, 1976). The hyphenation algorithm that is by far the most widely used now is due to Liang (Liang, 1983). Although this method is well-known now as the one used in TEX and its derivatives, the first version of TEX used a different, simpler method. Liang's method was used also in `troff` and `groff`, which were the main original competitors of TEX, and is part of many contemporary software products, supposedly including Microsoft Word. Any major improvement over Liang's method is therefore of considerable practical and commercial importance.

Over the years, various machine learning methods have been applied to the hyphenation task. However, none have achieved high accuracy. One paper that presents three different learning methods is (van den Bosch et al., 1995). The lowest per-letter test error rate reported is about 2%. Neural networks have been used, but also without great success. For example, the authors of (Kristensen and Langmyhr, 2001) found that the TEX method is a better choice for hyphenating Norwegian.

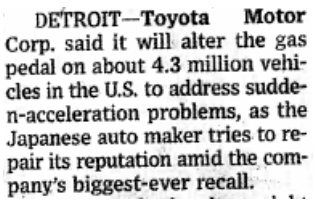
The highest accuracy achieved until now for the hyphenation task is by (Bartlett et al., 2008), who use a large-margin structured learning approach. Our work is similar, but was done fully independently. The accuracy we achieve is slightly higher: word-level accuracy of 96.33% compared to their 95.65% for English. Moreover, (Bartlett et al., 2008) do not address the issue that false positive hyphens are worse mistakes than false negative hyphens, which we address below. Also, they report that training on 14,000 examples requires about an hour, compared to 6.2 minutes for our method on 65,828 words. Perhaps more important for large-scale publishing applications, our system is about six times faster at syllabifying new text. The speed comparison is fair because the computer we use is slightly slower than the one they used.

Methods inspired by nonstatistical natural language processing research have also been proposed for the hyphenation task, in particular (Bouma, 2003; Tsalidis et al., 2004; Woestenburg, 2006; Haralambous, 2006). However, the methods for Dutch presented in (Bouma, 2003) were found to have worse performance than TEX. Moreover, our experimental results below show that the commercial software of (Woestenburg, 2006) allows hyphens incorrectly almost three times more often than TEX.

In general, a dictionary based approach has zero errors for words in the dictionary, but fails to work for words not included in it. A rule-based approach requires an expert to define manually the rules and exceptions for each language, which is laborious work. Furthermore, for languages such as English where hyphenation does not systematically follow general rules, such an approach does not have good results. A pattern-learning approach, like that of TEX, infers patterns from a training list of hyphenated words, and then uses these patterns to hyphenate text. Although useful patterns are learned automatically, both the TEX learning algorithm and the learned patterns must be hand-tuned to perform well (Liang, 1983).

Liang's method is implemented in a program named `PATGEN`, which takes as input a training set of hyphenated words, and outputs a collection of interacting hyphenation patterns. The standard pattern collections are named `hyphen.tex` for American English, `ukhyphen.tex` for British English, and `nehyph96.tex` for Dutch. The precise details of how different versions of TEX and LATEX use these pattern collections to do hyphenation in practice are unclear. At a minimum, current variants of TEX improve hyphenation accuracy by disallowing hyphens in the first and last two or three letters of every word, regardless of what the `PATGEN` patterns recommend.

Despite the success of Liang's method, incorrect hyphenations remain an issue with TEX and its current variants and competitors. For instance, incorrect hyphenations are common in the *Wall Street Journal*, which has the highest circulation of any newspaper in the U.S. An example is the hyphenation of the word "sudden" in this extract:


DETROIT—Toyota Motor Corp. said it will alter the gas pedal on about 4.3 million vehicles in the U.S. to address sudden-acceleration problems, as the Japanese auto maker tries to repair its reputation amid the company's biggest-ever recall.

It is the case that most hyphenation mistakes in the *Wall Street Journal* and other media are for proper nouns such as "Netflix" that do not appear in standard dictionaries, or in compound words such as "sudden-acceleration" above.

3 Conditional random fields

A linear-chain conditional random field (Lafferty et al., 2001) is a way to use a log-linear model for the sequence prediction task. We use the bar notation for sequences, so $\bar{x}$ means a sequence of variable length. Specifically, let $\bar{x}$ be a sequence of n letters and let $\bar{y}$ be a corresponding sequence of n tags. Define the log-linear model

$$p(\bar{y}|\bar{x};w) = \frac{1}{Z(\bar{x},w)} \exp \sum_j w_j F_j(\bar{x},\bar{y}).$$

The index j ranges over a large set of feature-functions. Each such function F_j is a sum along the output sequence for $i = 1$ to $i = n$:

$$F_j(\bar{x},\bar{y}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \bar{x}, i)$$

where each function f_j is a 0/1 indicator function that picks out specific values for neighboring tags y_{i-1} and y_i and a particular substring of $\bar{x}$. The denominator $Z(\bar{x},w)$ is a normalizing constant:

$$Z(\bar{x},w) = \sum_{\bar{y}} \exp \sum_j w_j F_j(\bar{x},\bar{y})$$

where the outer sum is over all possible labelings $\bar{y}$ of the input sequence $\bar{x}$. Training a CRF means finding a weight vector w that gives the best possible predictions

$$\bar{y}^* = \arg\max_{\bar{y}} p(\bar{y}|\bar{x};w)$$

for each training example $\bar{x}$.

The software we use as an implementation of conditional random fields is named CRF++ (Kudo, 2007). This implementation offers fast training since it uses L-BFGS (Nocedal and Wright, 1999), a state-of-the-art quasi-Newton method for large optimization problems. We adopt the default parameter settings of CRF++, so no development set or tuning set is needed in our work.

We define indicator functions f_j that depend on substrings of the input word, and on whether or not a hyphen is legal after the current and/or the previous letter. The substrings are of length 2 to 5, covering up to 4 letters to the left and right of the current letter. From all possible indicator functions we use only those that involve a substring that occurs at least once in the training data.

As an example, consider the word `hy-phen-ate`. For this word $\bar{x} = hyphenate$ and $\bar{y} = 010001000$. Suppose $i = 3$ so p is the current letter. Then exactly two functions f_j that depend on substrings of length 2 have value 1:

$$I(y_{i-1} = 1 \text{ and } y_i = 0 \text{ and } x_2x_3 = yp) = 1,$$
$$I(y_{i-1} = 1 \text{ and } y_i = 0 \text{ and } x_3x_4 = ph) = 1.$$

All other similar functions have value 0:

$$I(y_{i-1} = 1 \text{ and } y_i = 1 \text{ and } x_2x_3 = yp) = 0,$$
$$I(y_{i-1} = 1 \text{ and } y_i = 0 \text{ and } x_2x_3 = yq) = 0,$$

and so on. There are similar indicator functions for substrings up to length 5. In total, 2,916,942 different indicator functions involve a substring that appears at least once in the English dataset.

One finding of our work is that it is preferable to use a large number of low-level features, that is patterns of specific letters, rather than a smaller number of higher-level features such as consonant-vowel patterns. This finding is consistent with an emerging general lesson about many natural language processing tasks: the best performance is achieved with models that are discriminative, that are trained on as large a dataset as possible, and that have a very large number of parameters but are regularized (Halevy et al., 2009).

When evaluating the performance of a hyphenation algorithm, one should not just count how many words are hyphenated in exactly the same way as in a reference dictionary. One should also measure separately how many legal hyphens are actually predicted, versus how many predicted hyphens are in fact not legal. Errors of the second type are false positives. For any hyphenation

method, a false positive hyphen is a more serious mistake than a false negative hyphen, i.e. a hyphen allowed by the lexicon that the method fails to identify. The standard Viterbi algorithm for making predictions from a trained CRF is not tuned to minimize false positives. To address this difficulty, we use the forward-backward algorithm (Sha and Pereira, 2003; Culotta and McCallum, 2004) to estimate separately for each position the probability of a hyphen at that position. Then, we only allow a hyphen if this probability is over a high threshold such as 0.9.

Each hyphenation corresponds to one path through a graph that defines all 2^{k-1} hyphenations that are possible for a word of length k. The overall probability of a hyphen at any given location is the sum of the weights of all paths that do have a hyphen at this position, divided by the sum of the weights of all paths. The forward-backward algorithm uses the sum operator to compute the weight of a set of paths, instead of the max operator to compute the weight of a single highest-weight path. In order to compute the weight of all paths that contain a hyphen at a specific location, weight 0 is assigned to all paths that do not have a hyphen at this location.

4 Dataset creation

We start with the lexicon for English published by the Dutch Centre for Lexical Information at `http://www.mpi.nl/world/celex`. We download all English word forms with legal hyphenation points indicated by hyphens. These include plurals of nouns, conjugated forms of verbs, and compound words such as "off-line". We separate the components of compound words and phrases, leading to 204,466 words, of which 68,744 are unique. In order to eliminate abbreviations and proper names which may not be English, we remove all words that are not fully lower-case. In particular, we exclude words that contain capital letters, apostrophes, and/or periods. This leaves 66,001 words.

Among these words, 86 have two different hyphenations, and one has three hyphenations. For most of the 86 words with alternative hyphenations, these alternatives exist because different meanings of the words have different pronunciations, and the different pronunciations have different boundaries between syllables. This fact implies that no algorithm that operates on words in isolation can be a complete solution for the hyphenation task.[1]

We exclude the few words that have two or more different hyphenations from the dataset. Finally, we obtain 65,828 spellings. These have 550,290 letters and 111,228 hyphens, so the average is 8.36 letters and 1.69 hyphens per word. Informal inspection suggests that the 65,828 spellings contain no mistakes. However, about 1000 words follow British as opposed to American spelling.

The Dutch dataset of 293,681 words is created following the same procedure as for the English dataset, except that all entries from CELEX that are compound words containing dashes are discarded instead of being split into parts, since many of these are not in fact Dutch words.[2]

5 Experimental design

We use ten-fold cross validation for the experiments. In order to measure accuracy, we compute the confusion matrix for each method, and from this we compute error rates. We report both word-level and letter-level error rates. The word-level error rate is the fraction of words on which a method makes at least one mistake. The letter-level error rate is the fraction of letters for which the method predicts incorrectly whether or not a hyphen is legal after this letter. Table 1 explains the terminology that we use in presenting our results. Precision, recall, and F1 can be computed easily from the reported confusion matrices.

As an implementation of Liang's method we use TEX Hyphenator in Java software available at `http://texhyphj.sourceforge.net`. We evaluate this algorithm on our entire English and Dutch datasets using the appropriate language pattern files, and not allowing a hyphen to be placed between the first `lefthyphenmin` and last `righthyphenmin` letters of each word. For

[1] The single word with more than two alternative hyphenations is "invalid" whose three hyphenations are `in-va-lid in-val-id` and `in-valid`. Interestingly, the Merriam–Webster online dictionary also gives three hyphenations for this word, but not the same ones: `in-va-lid in-val-id invalid`. The American Heritage dictionary agrees with Merriam-Webster. The disagreement illustrates that there is a certain irreducible ambiguity or subjectivity concerning the correctness of hyphenations.

[2] Our English and Dutch datasets are available for other researchers and practitioners to use at `http://www.cs.ucsd.edu/users/elkan/hyphenation`. Previously a similar but smaller CELEX-based English dataset was created by (van den Bosch et al., 1995), but that dataset is not available online currently.

Abbr	Name	Description
TP	true positives	#hyphens predicted correctly
FP	false positives	#hyphens predicted incorrectly
TN	true negatives	#hyphens correctly not predicted
FN	false negatives	#hyphens failed to be predicted
owe	overall word-level errors	#words with at least one FP or FN
swe	serious word-level errors	#words with at least one FP
ower	overall word-level error rate	owe / (total #words)
swer	serious word-level error rate	swe / (total #words)
oler	overall letter-level error rate	(FP+FN) / (TP+TN+FP+FN)
sler	serious letter-level error rate	FP / (TP+TN+FP+FN)

Table 1: Alternative measures of accuracy. TP, TN, FP, and FN are computed by summing over the test sets of each fold of cross-validation.

English the default values are 2 and 3 respectively. For Dutch the default values are both 2.

The hyphenation patterns used by TeXHyphenator, which are those currently used by essentially all variants of TEX, may not be optimal for our new English and Dutch datasets. Therefore, we also do experiments with the PATGEN tool (Liang and Breitenlohner, 2008). These are learning experiments so we also use ten-fold cross validation in the same way as with CRF++. Specifically, we create a pattern file from 90% of the dataset using PATGEN, and then hyphenate the remaining 10% of the dataset using Liang's algorithm and the learned pattern file.

The PATGEN tool has many user-settable parameters. As is the case with many machine learning methods, no strong guidance is available for choosing values for these parameters. For English we use the parameters reported in (Liang, 1983). For Dutch we use the parameters reported in (Tutelaers, 1999). Preliminary informal experiments found that these parameters work better than alternatives. We also disallow hyphens in the first two letters of every word, and the last three letters for English, or last two for Dutch.

We also evaluate the TALO commercial software (Woestenburg, 2006). We know of one other commercial hyphenation application, which is named Dashes.[3] Unfortunately we do not have access to it for evaluation. We also cannot do a precise comparison with the method of (Bartlett et al., 2008). We do know that their training set was also derived from CELEX, and their maximum reported accuracy is slightly lower. Specifically, for English our word-level accuracy ("ower") is 96.33% while their best ("WA") is 95.65%.

[3] http://www.circlenoetics.com/dashes.aspx

6 Experimental results

In Table 2 and Table 3 we report the performance of the different methods on the English and Dutch datasets respectively. Figure 1 shows how the error rate is affected by increasing the CRF probability threshold for each language.

Figure 1 shows confidence intervals for the error rates. These are computed as follows. For a single Bernoulli trial the mean is p and the variance is $p(1-p)$. If N such trials are taken, then the observed success rate $f = S/N$ is a random variable with mean p and variance $p(1-p)/N$. For large N, the distribution of the random variable f approaches the normal distribution. Hence we can derive a confidence interval for p using the formula

$$Pr[-z \leq \frac{f-p}{\sqrt{p(1-p)/N}} \leq z] = c$$

where for a 95% confidence interval, i.e. for $c = 0.95$, we set $z = 1.96$. All differences between rows in Table 2 are significant, with one exception: the serious error rates for PATGEN and TALO are not statistically significantly different. A similar conclusion applies to Table 3.

For the English language, the CRF using the Viterbi path has overall error rate of 0.84%, compared to 6.81% for the TEX algorithm using American English patterns, which is eight times worse. However, the serious error rate for the CRF is less good: 0.41% compared to 0.24%. This weakness is remedied by predicting that a hyphen is allowable only if it has high probability. Figure 1 shows that the CRF can use a probability threshold up to 0.99, and still have lower overall error rate than the TEX algorithm. Fixing the probability threshold at 0.99, the CRF serious error rate is 0.04% (224 false positives) compared to 0.24% (1343 false positives) for the TEX algorithm.

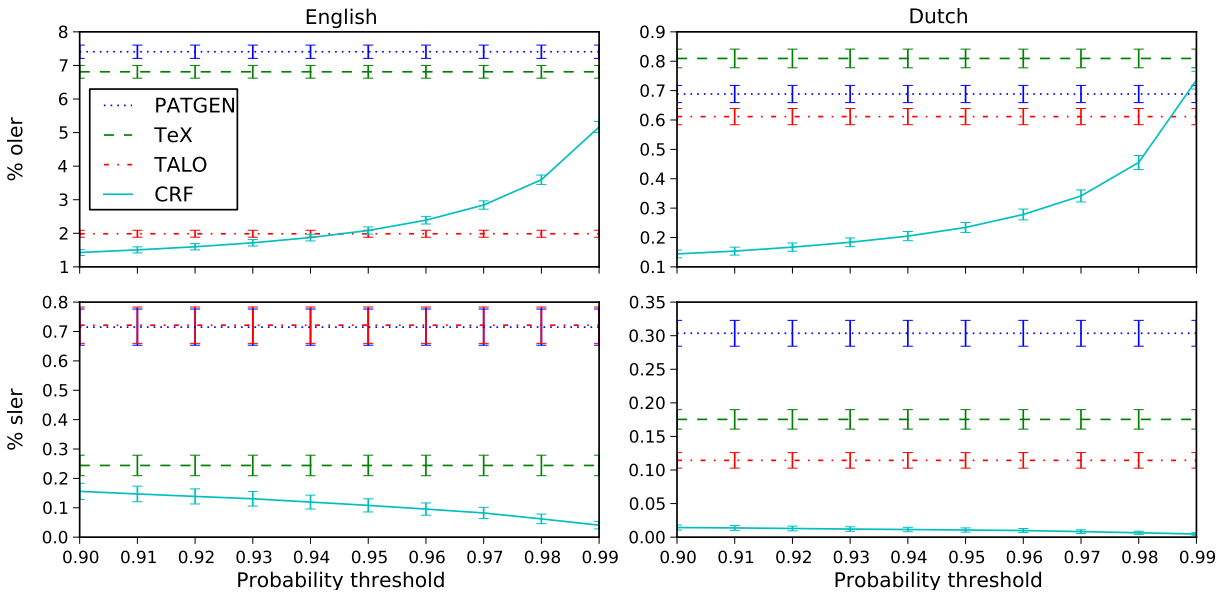


Figure 1: Total letter-level error rate and serious letter-level error rate for different values of threshold for the CRF. The left subfigures are for the English dataset, while the right ones are for the Dutch dataset. The TALO and PATGEN lines are almost identical in the bottom left subfigure.

Method	TP	FP	TN	FN	owe	swe	% ower	% swer	% oler	% sler
Place no hyphen	0	0	439062	111228	57541	0	87.41	0.00	20.21	0.00
TEX (hyphen.tex)	75093	1343	437719	36135	30337	1311	46.09	1.99	6.81	0.24
TEX (ukhyphen.tex)	70307	13872	425190	40921	31337	11794	47.60	17.92	9.96	2.52
TALO	104266	3970	435092	6962	7213	3766	10.96	5.72	1.99	0.72
PATGEN	74397	3934	435128	36831	32348	3803	49.14	5.78	7.41	0.71
CRF	**108859**	2253	436809	**2369**	**2413**	2080	**3.67**	3.16	**0.84**	0.41
CRF (threshold = 0.99)	83021	**224**	**438838**	28207	22992	**221**	34.93	**0.34**	5.17	**0.04**

Table 2: Performance on the English dataset.

Method	TP	FP	TN	FN	owe	swe	% ower	% swer	% oler	% sler
Place no hyphen	0	0	2438913	742965	287484	0	97.89	0.00	23.35	0.00
TEX (nehyph96.tex)	722789	5580	2433333	20176	20730	5476	7.06	1.86	0.81	0.18
TALO	727145	3638	2435275	15820	16346	3596	5.57	1.22	0.61	0.11
PATGEN	730720	9660	2429253	12245	20318	9609	6.92	3.27	0.69	0.30
CRF	**741796**	1230	2437683	**1169**	**1443**	1207	**0.49**	0.41	**0.08**	0.04
CRF (threshold = 0.99)	719710	**149**	**2438764**	23255	22067	**146**	7.51	**0.05**	0.74	**0.00**

Table 3: Performance on the Dutch dataset.

Method	TP	FP	TN	FN	owe	swe	% ower	% swer	% oler	% sler
PATGEN	70357	6763	432299	40871	35013	6389	53.19	9.71	8.66	1.23
CRF	**104487**	6518	432544	**6741**	**6527**	5842	**9.92**	8.87	**2.41**	1.18
CRF (threshold = 0.99)	75651	**654**	**438408**	35577	27620	**625**	41.96	**0.95**	6.58	**0.12**

Table 4: Performance on the English dataset (10-fold cross validation dividing by stem).

Method	TP	FP	TN	FN	owe	swe	% ower	% swer	% oler	% sler
PATGEN	727306	13204	2425709	15659	25363	13030	8.64	4.44	0.91	0.41
CRF	**740331**	2670	2436243	**2634**	**3066**	2630	**1.04**	0.90	**0.17**	0.08
CRF (threshold = 0.99)	716596	**383**	**2438530**	26369	24934	**373**	8.49	**0.13**	0.84	**0.01**

Table 5: Performance on the Dutch dataset (10-fold cross validation dividing by stem).

Method	TP	FP	TN	FN	owe	swe	% ower	% swer	% oler	% sler
TEX	2711	43	21433	1420	1325	43	33.13	1.08	5.71	0.17
PATGEN	2590	113	21363	1541	1466	113	36.65	2.83	6.46	0.44
CRF	**4129**	2	21474	**2**	**2**	2	**0.05**	0.05	**0.02**	0.01
CRF (threshold = 0.9)	4065	**0**	**21476**	66	63	**0**	1.58	**0.00**	0.26	**0.00**

Table 6: Performance on the 4000 most frequent English words.

For the English language, TALO yields overall error rate 1.99% with serious error rate 0.72%, so the standard CRF using the Viterbi path is better on both measures. The dominance of the CRF method can be increased further by using a probability threshold. Figure 1 shows that the CRF can use a probability threshold up to 0.94, and still have lower overall error rate than TALO. Using this threshold, the CRF serious error rate is 0.12% (657 false positives) compared to 0.72% (3970 false positives) for TALO.

For the Dutch language, the standard CRF using the Viterbi path has overall error rate 0.08%, compared to 0.81% for the TEX algorithm. The serious error rate for the CRF is 0.04% while for TEX it is 0.18%. Figure 1 shows that any probability threshold for the CRF of 0.99 or below yields lower error rates than the TEX algorithm. Using the threshold 0.99, the CRF has serious error rate only 0.005%.

For the Dutch language, the TALO method has overall error rate 0.61%. The serious error rate for TALO is 0.11%. The CRF dominance can again be increased via a high probability threshold. Figure 1 shows that this threshold can range up to 0.98, and still give lower overall error rate than TALO. Using the 0.98 threshold, the CRF has serious error rate 0.006% (206 false positives); in comparison the serious error rate of TALO is 0.11% (3638 false positives).

For both languages, PATGEN has higher serious letter-level and word-level error rates than TEX using the existing pattern files. This is expected since the pattern collections included in TEX distributions have been tuned over the years to minimize objectionable errors. The difference is especially pronounced for American English, for which the standard pattern collection has been manually improved over more than two decades by many people (Beeton, 2002). Initially, Liang optimized this pattern collection extensively by upweighting the most common words and by iteratively adding exception words found by testing the algorithm against a large dictionary from an unknown publisher (Liang, 1983).

One can tune PATGEN to yield either better overall error rate, or better serious error rate, but not both simultaneously, compared to the TEX algorithm using the existing pattern files for both languages. For the English dataset, if we use Liang's parameters for PATGEN as reported in (Sojka and Sevecek, 1995), we obtain overall error rate of 6.05% and serious error rate of 0.85%. It is possible that the specific patterns used in TEX implementations today have been tuned by hand to be better than anything the PATGEN software is capable of.

7 Additional experiments

This section presents empirical results following two experimental designs that are less standard, but that may be more appropriate for the hyphenation task.

First, the experimental design used above has an issue shared by many CELEX-based tagging or transduction evaluations: words are randomly divided into training and test sets without being grouped by stem. This means that a method can get credit for hyphenating "accents" correctly, when "accent" appears in the training data. Therefore, we do further experiments where the folds for evaluation are divided by stem, and not by word; that is, all versions of a base form of a word appear in the same fold. Stemming uses the English and Dutch versions of the Porter stemmer (Porter, 1980).[4] The 65,828 English words in our dictionary produce 27,100 unique stems, while the 293,681 Dutch words produce 169,693 unique stems. The results of these experiments are shown in Tables 4 and 5.

The main evaluation in the previous section is based on a list of unique words, which means that in the results each word is equally weighted. Because cross validation is applied, errors are always measured on testing subsets that are disjoint from the corresponding training subsets. Hence, the accuracy achieved can be interpreted as the performance expected when hyphenating unknown words, i.e. rare future words.

However, in real documents common words appear repeatedly. Therefore, the second less-standard experimental design for which we report results restricts attention to the most common English words. Specifically, we consider the top 4000 words that make up about three quarters of all word appearances in the American National Corpus, which consists of 18,300,430 words from written texts of all genres.[5] From the 4,471 most

[4] Available at http://snowball.tartarus.org/. A preferable alternative might be to use the information about the lemmas of words available directly in CELEX.

[5] Available at americannationalcorpus.org/SecondRelease/data/ANC-written-count.txt

frequent words in this list, if we omit the words not in our dataset of 89,019 hyphenated English words from CELEX, we get 4,000 words. The words that are omitted are proper names, contractions, incomplete words containing apostrophes, and abbreviations such as DNA. These 4,000 most frequent words make up 74.93% of the whole corpus.

We evaluate the following methods on the 4000 words: Liang's method using the American patterns file `hyphen.tex`, Liang's method using the patterns derived from `PATGEN` when trained on the whole English dataset, our CRF trained on the whole English dataset, and the same CRF with a probability threshold of 0.9. Results are shown in Table 6. In summary, TEX and `PATGEN` make serious errors on 43 and 113 of the 4000 words, respectively. With a threshold of 0.9, the CRF approach makes zero serious errors on these words.

8 Timings

Table 7 shows the speed of the alternative methods for the English dataset. The column "Features/Patterns" in the table reports the number of feature-functions used for the CRF, or the number of patterns used for the TEX algorithm. Overall, the CRF approach is about ten times slower than the TEX algorithm, but its performance is still acceptable on a standard personal computer. All experiments use a machine having a Pentium 4 CPU at 3.20GHz and 2GB memory. Moreover, informal experiments show that CRF training would be about eight times faster if we used CRFSGD rather than CRF++ (Bottou, 2008).

From a theoretical perspective, both methods have almost-constant time complexity per word if they are implemented using appropriate data structures. In TEX, hyphenation patterns are stored in a data structure that is a variant of a trie. The CRF software uses other data structures and optimizations that allow a word to be hyphenated in time that is almost independent of the number of feature-functions used.

9 Conclusions

Finding allowable places in words to insert hyphens is a real-world problem that is still not fully solved in practice. The main contribution of this paper is a hyphenation method that is clearly more accurate than the currently used Knuth/Liang method. The new method is an application of CRFs, which are a major advance of recent years in machine learning. We hope that the method proposed here is adopted in practice, since the number of serious errors that it makes is about a sixfold improvement over what is currently in use. A second contribution of this paper is to provide training sets for hyphenation in English and Dutch, so other researchers can, we hope, soon invent even more accurate methods. A third contribution of our work is a demonstration that current CRF methods can be used straightforwardly for an important application and outperform state-of-the-art commercial and open-source software; we hope that this demonstration accelerates the widespread use of CRFs.

Method	Features/ Patterns	Training time (s)	Testing time (s)	Speed (ms/word)
CRF	2916942	372.67	25.386	0.386
TEX (us)	4447	-	2.749	0.042
`PATGEN`	4488	33.402	2.889	0.044
TALO	-	-	8.400	0.128

Table 7: Timings for the English dataset (training and testing on the whole dataset that consists of 65,828 words).

References

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2008. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. *Proceedings of ACL-08: HLT*, pages 568–576.

Barbara Beeton. 2002. Hyphenation exception log. *TUGboat*, 23(3).

Léon Bottou. 2008. Stochastic gradient CRF software CRFSGD. Available at `http://leon.bottou.org/projects/sgd`.

Gosse Bouma. 2003. Finite state methods for hyphenation. *Natural Language Engineering*, 9(1):5–20, March.

Aron Culotta and Andrew McCallum. 2004. Confidence Estimation for Information Extraction. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 109–112, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Fred J. Damerau. 1964. Automatic Hyphenation Scheme. U.S. patent 3537076 filed June 17, 1964, issued October 1970.

Gordon D. Friedlander. 1968. Automation comes to the printing and publishing industry. *IEEE Spectrum*, 5:48–62, April.

Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12.

Yannis Haralambous. 2006. New hyphenation techniques in Ω_2. *TUGboat*, 27:98–103.

Steven L. Huyser. 1976. AUTO-MA-TIC WORD DIVI-SION. *SIGDOC Asterisk Journal of Computer Documentation*, 3(5):9–10.

Timo Jarvi. 2009. Computerized Typesetting and Other New Applications in a Publishing House. In *History of Nordic Computing 2*, pages 230–237. Springer.

Terje Kristensen and Dag Langmyhr. 2001. Two regimes of computer hyphenation–a comparison. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 1532–1535.

Taku Kudo, 2007. *CRF++: Yet Another CRF Toolkit*. Version 0.5 available at `http://crfpp.sourceforge.net/`.

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

Franklin M. Liang and Peter Breitenlohner, 2008. *PATtern GENeration Program for the TEX82 Hyphenator*. Electronic documentation of PATGEN program version 2.3 from web2c distribution on CTAN, retrieved 2008.

Franklin M. Liang. 1983. *Word Hy-phen-a-tion by Com-put-er*. Ph.D. thesis, Stanford University.

Jorge Nocedal and Stephen J. Wright. 1999. Limited memory BFGS. In *Numerical Optimization*, pages 222–247. Springer.

Wolfgang A. Ocker. 1971. A program to hyphenate English words. *IEEE Transactions on Engineering, Writing and Speech*, 14(2):53–59, June.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Terrence J. Sejnowski and Charles R. Rosenberg, 1988. *NETtalk: A parallel network that learns to read aloud*, pages 661–672. MIT Press, Cambridge, MA, USA.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141.

Petr Sojka and Pavel Sevecek. 1995. Hyphenation in TEX–Quo Vadis? *TUGboat*, 16(3):280–289.

Christos Tsalidis, Giorgos Orphanos, Anna Iordanidou, and Aristides Vagelatos. 2004. Proofing Tools Technology at Neurosoft S.A. *ArXiv Computer Science e-prints*, (cs/0408059), August.

P.T.H. Tutelaers, 1999. *Afbreken in TEX, hoe werkt dat nou?* Available at `ftp://ftp.tue.nl/pub/tex/afbreken/`.

Antal van den Bosch, Ton Weijters, Jaap Van Den Herik, and Walter Daelemans. 1995. The profit of learning exceptions. In *Proceedings of the 5th Belgian-Dutch Conference on Machine Learning (BENELEARN)*, pages 118–126.

Jaap C. Woestenburg, 2006. **TALO's Language Technology*, November. Available at `http://www.talo.nl/talo/download/documents/Language_Book.pdf`.

Enhanced word decomposition by calibrating the decision threshold of probabilistic models and using a model ensemble

Sebastian Spiegler
Intelligent Systems Laboratory,
University of Bristol, U.K.
spiegler@cs.bris.ac.uk

Peter A. Flach
Intelligent Systems Laboratory,
University of Bristol, U.K.
peter.flach@bristol.ac.uk

Abstract

This paper demonstrates that the use of ensemble methods and carefully calibrating the decision threshold can significantly improve the performance of machine learning methods for morphological word decomposition. We employ two algorithms which come from a family of generative probabilistic models. The models consider segment boundaries as hidden variables and include probabilities for letter transitions within segments. The advantage of this model family is that it can learn from small datasets and easily generalises to larger datasets. The first algorithm PROMODES, which participated in the Morpho Challenge 2009 (an international competition for unsupervised morphological analysis) employs a lower order model whereas the second algorithm PROMODES-H is a novel development of the first using a higher order model. We present the mathematical description for both algorithms, conduct experiments on the morphologically rich language Zulu and compare characteristics of both algorithms based on the experimental results.

1 Introduction

Words are often considered as the smallest unit of a language when examining the grammatical structure or the meaning of sentences, referred to as syntax and semantics, however, words themselves possess an internal structure denominated by the term *word morphology*. It is worthwhile studying this internal structure since a language description using its morphological formation is more compact and complete than listing all possible words. This study is called *morphological analysis*. According to Goldsmith (2009) four tasks are assigned to morphological analysis: *word decomposition* into morphemes, building *morpheme dictionaries*, defining *morphosyntactical rules* which state how morphemes can be combined to valid words and defining *morphophonological rules* that specify phonological changes morphemes undergo when they are combined to words. Results of morphological analysis are applied in speech synthesis (Sproat, 1996) and recognition (Hirsimaki et al., 2006), machine translation (Amtrup, 2003) and information retrieval (Kettunen, 2009).

1.1 Background

In the past years, there has been a lot of interest and activity in the development of algorithms for morphological analysis. All these approaches have in common that they build a *morphological model* which is then applied to analyse words. Models are constructed using *rule-based methods* (Mooney and Califf, 1996; Muggleton and Bain, 1999), *connectionist methods* (Rumelhart and McClelland, 1986; Gasser, 1994) or *statistical* or *probabilistic* methods (Harris, 1955; Hafer and Weiss, 1974). Another way of classifying approaches is based on the *learning aspect* during the construction of the morphological model. If the data for training the model has the same structure as the desired output of the morphological analysis, in other words, if a morphological model is learnt from labelled data, the algorithm is classified under *supervised learning*. An example for a supervised algorithm is given by Oflazer et al. (2001). If the input data has no information towards the desired output of the analysis, the algorithm uses *unsupervised learning*. Unsupervised algorithms for morphological analysis are Linguistica (Goldsmith, 2001), Morfessor (Creutz, 2006) and Paramor (Monson, 2008). *Minimally or semi-supervised algorithms* are provided with partial information during the learning process. This

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 375–383,
Uppsala, Sweden, 11-16 July 2010.

has been done, for instance, by Shalonova et al. (2009) who provided stems in addition to a word list in order to find multiple pre- and suffixes. A comparison of different levels of supervision for morphology learning on Zulu has been carried out by Spiegler et al. (2008).

Our two algorithms, PROMODES and PROMODES-H, perform *word decomposition* and are based on *probabilistic methods* by incorporating a probabilistic generative model.[1] Their parameters can be estimated from either labelled data, using maximum likelihood estimates, or from unlabelled data by expectation maximization[2] which makes them either *supervised* or *unsupervised* algorithms.

The purpose of this paper is an analysis of the underlying probabilistic models and the types of errors committed by each one. Furthermore, it is investigated how the decision threshold can be calibrated and a model ensemble is tested.

The remainder is structured as follows. In Section 2 we introduce the probabilistic generative process and show in Sections 2.1 and 2.2 how we incorporate this process in PROMODES and PROMODES-H. We start our experiments with examining the learning behaviour of the algorithms in 3.1. Subsequently, we perform a position-wise comparison of predictions in 3.2, show how we find a better decision threshold for placing morpheme boundaries in 3.3 and combine both algorithms using a model ensemble to leverage individual strengths in 3.4. In 3.5 we examine how the single algorithms contribute to the result of the ensemble. In Section 4 we will compare our approaches to related work and in Section 5 we will draw our conclusions.

2 Probabilistic generative model

Intuitively, we could say that our models describe the process of word generation from the left to the right by alternately using two dice, the first for deciding whether to place a morpheme boundary in the current word position and the second to get a corresponding letter transition. We are trying to reverse this process in order to find the underlying sequence of tosses which determine the morpheme boundaries. We are applying the notion of a *probabilistic generative process* consisting of words as observed variables X and their hidden segmentation as latent variables Y. If a generative model is fully parameterised it can be reversed to find the underlying word decomposition by forming the conditional probability distribution $Pr(Y|X)$.

Let us first define the model-independent components. A given *word* $w_j \in W$ with $1 \leq j \leq |W|$ consists of n letters and has $m = n-1$ positions for inserting boundaries. A word's segmentation is depicted as a *boundary vector* $b_j = (b_{j1}, \ldots, b_{jm})$ consisting of *boundary values* $b_{ji} \in \{0,1\}$ with $1 \leq i \leq m$ which disclose whether or not a boundary is placed in position i. A *letter* $l_{j,i\text{-}1}$ precedes the position i in w_j and a letter l_{ji} follows it. Both letters $l_{j,i\text{-}1}$ and l_{ji} are part of an alphabet. Furthermore, we introduce a *letter transition* t_{ji} which goes from $l_{j,i\text{-}1}$ to l_{ji}.

2.1 PROMODES

PROMODES is based on a zero-order model for boundaries b_{ji} and on a first-order model for letter transitions t_{ji}. It describes a word's segmentation by its morpheme boundaries and resulting letter transitions within morphemes. A boundary vector b_j is found by evaluating each position i with

$$\begin{aligned} &\arg\max_{b_{ji}} Pr(b_{ji}|t_{ji}) = \\ &\arg\max_{b_{ji}} Pr(b_{ji})Pr(t_{ji}|b_{ji}) \quad . \end{aligned} \tag{1}$$

The first component of the equation above is the *probability distribution over non-/boundaries* $Pr(b_{ji})$. We assume that a boundary in i is inserted independently from other boundaries (zero-order) and the graphemic representation of the word, however, is conditioned on the length of the word m_j which means that the probability distribution is in fact $Pr(b_{ji}|m_j)$. We guarantee $\sum_{r=0}^{1} Pr(b_{ji}{=}r|m_j) = 1$. To simplify the notation in later explanations, we will refer to $Pr(b_{ji}|m_j)$ as $Pr(b_{ji})$.

The second component is the *letter transition probability distribution* $Pr(t_{ji}|b_{ji})$. We suppose a first-order Markov chain consisting of transitions t_{ji} from letter $l_{j,i\text{-}1} \in A_{\mathscr{B}}$ to letter $l_{ji} \in A$ where A is a regular letter alphabet and $A_{\mathscr{B}}{=}A \cup \{\mathscr{B}\}$ includes $\mathscr{B}$ as an abstract morpheme start symbol which can occur in $l_{j,i\text{-}1}$. For instance, the suffix '*s*' of the verb form *gets*, marking *3rd person singular*, would be modelled as $\mathscr{B} \rightarrow s$ whereas a morpheme internal transition could be $g \rightarrow e$. We

[1] PROMODES stands for PRObabilistic MOdel for different DEgrees of Supervision. The H of PROMODES-H refers to Higher order.

[2] In (Spiegler et al., 2009; Spiegler et al., 2010a) we have presented an unsupervised version of PROMODES.

guarantee $\sum_{l_{ji} \in A} Pr(t_{ji}|b_{ji})=1$ with t_{ji} being a transition from a certain $l_{j,i-1} \in A_{\mathscr{B}}$ to l_{ji}. The advantage of the model is that instead of evaluating an exponential number of possible segmentations (2^m), the best segmentation $b^*_j=(b^*_{j1},\ldots,b^*_{jm})$ is found with $2m$ position-wise evaluations using

$$b^*_{ji} = \arg\max_{b_{ji}} Pr(b_{ji}|t_{ji}) \quad (2)$$

$$= \begin{cases} 1, \text{ if } Pr(b_{ji}=1)Pr(t_{ji}|b_{ji}=1) \\ \quad > Pr(b_{ji}=0)Pr(t_{ji}|b_{ji}=0) \\ 0, \text{ otherwise } . \end{cases}$$

The simplifying assumptions made, however, reduce the expressive power of the model by not allowing any dependencies on preceding boundaries or letters. This can lead to over-segmentation and therefore influences the performance of PROMODES. For this reason, we have extended the model which led to PROMODES-H, a higher-order probabilistic model.

2.2 PROMODES-H

In contrast to the original PROMODES model, we also consider the boundary value $b_{j,i\text{-}1}$ and modify our transition assumptions for PROMODES-H in such a way that the new algorithm applies a first-order boundary model and a second-order transition model. A transition t_{ji} is now defined as a transition from an abstract symbol in $l_{j,i\text{-}1} \in \{\mathscr{N},\mathscr{B}\}$ to a letter in $l_{ji} \in A$. The abstract symbol is $\mathscr{N}$ or $\mathscr{B}$ depending on whether b_{ji} is 0 or 1. This holds equivalently for letter transitions $t_{j,i\text{-}1}$. The suffix of our previous example *gets* would be modelled $\mathscr{N} \rightarrow t \rightarrow \mathscr{B} \rightarrow s$.

Our boundary vector b_j is then constructed from

$$\arg\max_{b_{ji}} Pr(b_{ji}|t_{ji},t_{j,i\text{-}1},b_{j,i\text{-}1}) = \quad (3)$$
$$\arg\max_{b_{ji}} Pr(b_{ji}|b_{j,i\text{-}1})Pr(t_{ji}|b_{ji},t_{j,i\text{-}1},b_{j,i\text{-}1}) \ .$$

The first component, the *probability distribution over non-/boundaries* $Pr(b_{ji}|b_{j,i\text{-}1})$, satisfies $\sum_{r=0}^{1} Pr(b_{ji}=r|b_{j,i\text{-}1})=1$ with $b_{j,i\text{-}1}, b_{ji} \in \{0,1\}$. As for PROMODES, $Pr(b_{ji}|b_{j,i\text{-}1})$ is shorthand for $Pr(b_{ji}|b_{j,i\text{-}1},m_j)$. The second component, the *letter transition probability distribution* $Pr(t_{ji}|b_{ji},b_{j,i\text{-}1})$, fulfils $\sum_{l_{ji} \in A} Pr(t_{ji}|b_{ji},t_{j,i\text{-}1},b_{j,i\text{-}1})=1$ with t_{ji} being a transition from a certain $l_{j,i-1} \in A_{\mathscr{B}}$ to l_{ji}. Once again, we find the word's best segmentation b^*_j in $2m$ evaluations with

$$b^*_{ji} = \arg\max_{b_{ji}} Pr(b_{ji}|t_{ji},t_{j,i\text{-}1},b_{j,i\text{-}1}) = \quad (4)$$

$$\begin{cases} 1, \text{ if } Pr(b_{ji}=1|b_{j,i\text{-}1})Pr(t_{ji}|b_{ji}=1,t_{j,i\text{-}1},b_{j,i\text{-}1}) \\ \quad > Pr(b_{ji}=0|b_{j,i\text{-}1})Pr(t_{ji}|b_{ji}=0,t_{j,i\text{-}1},b_{j,i\text{-}1}) \\ 0, \text{ otherwise } . \end{cases}$$

We will show in the experimental results that increasing the memory of the algorithm by looking at $b_{j,i-1}$ leads to a better performance.

3 Experiments and Results

In the Morpho Challenge 2009, PROMODES achieved competitive results on Finnish, Turkish, English and German – and scored highest on non-vowelized and vowelized Arabic compared to 9 other algorithms (Kurimo et al., 2009). For the experiments described below, we chose the South African language *Zulu* since our research work mainly aims at creating morphological resources for under-resourced indigenous languages. Zulu is an *agglutinative language* with a complex morphology where multiple prefixes and suffixes contribute to a word's meaning. Nevertheless, it seems that segment boundaries are more likely in certain word positions. The PROMODES family harnesses this characteristic in combination with describing morphemes by letter transitions. From the *Ukwabelana corpus* (Spiegler et al., 2010b) we sampled 2500 Zulu words with a single segmentation each.

3.1 Learning with increasing experience

In our first experiment we applied 10-fold cross-validation on datasets ranging from 500 to 2500 words with the goal of measuring how the learning improves with increasing experience in terms of training set size. We want to remind the reader that our two algorithms are aimed at small datasets.

We randomly split each dataset into 10 subsets where each subset was a test set and the corresponding 9 remaining sets were merged to a training set. We kept the labels of the training set to determine model parameters through maximum likelihood estimates and applied each model to the test set from which we had removed the answer keys. We compared results on the test set against the ground truth by counting true positive (TP), false positive (FP), true negative (TN) and

false negative (FN) *morpheme boundary* predictions. Counts were summarised using precision[3], recall[4] and f-measure[5], as shown in Table 1.

Data	Precision	Recall	F-measure
500	0.7127±0.0418	0.3500±0.0272	0.4687±0.0284
1000	0.7435±0.0556	0.3350±0.0197	0.4614±0.0250
1500	0.7460±0.0529	0.3160±0.0150	0.4435±0.0206
2000	0.7504±0.0235	0.3068±0.0141	0.4354±0.0168
2500	0.7557±0.0356	0.3045±0.0138	0.4337±0.0163

(a) PROMODES

Data	Precision	Recall	F-measure
500	0.6983±0.0511	0.4938±0.0404	0.5776±0.0395
1000	0.6865±0.0298	0.5177±0.0177	0.5901±0.0205
1500	0.6952±0.0308	0.5376±0.0197	0.6058±0.0173
2000	0.7008±0.0140	0.5316±0.0146	0.6044±0.0110
2500	0.6941±0.0184	0.5396±0.0218	0.6068±0.0151

(b) PROMODES-H

Table 1: 10-fold cross-validation on Zulu.

For PROMODES we can see in Table 1a that the precision increases slightly from 0.7127 to 0.7557 whereas the recall decreases from 0.3500 to 0.3045 going from dataset size 500 to 2500. This suggests that to some extent fewer morpheme boundaries are discovered but the ones which are found are more likely to be correct. We believe that this effect is caused by the limited memory of the model which uses order zero for the occurrence of a boundary and order one for letter transitions. It seems that the model gets quickly saturated in terms of incorporating new information and therefore precision and recall do not drastically change for increasing dataset sizes. In Table 1b we show results for PROMODES-H. Across the datasets precision stays comparatively constant around a mean of 0.6949 whereas the recall increases from 0.4938 to 0.5396. Compared to PROMODES we observe an increase in recall between 0.1438 and 0.2351 at a cost of a decrease in precision between 0.0144 and 0.0616.

Since both algorithms show different behaviour with increasing experience and PROMODES-H yields a higher f-measure across all datasets, we will investigate in the next experiments how these differences manifest themselves at the boundary level.

[3] $precision = \frac{TP}{TP+FP}$.
[4] $recall = \frac{TP}{TP+FN}$.
[5] $f\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision+recall}$.

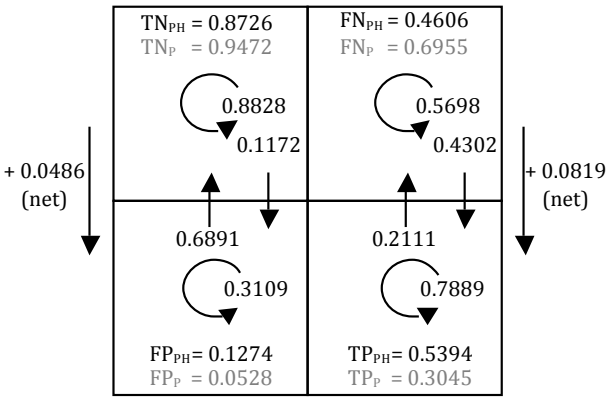


Figure 1: Contingency table for PROMODES [grey with subscript *P*] and PROMODES-H [black with subscript *PH*] results including gross and net changes of PROMODES-H.

3.2 Position-wise comparison of algorithmic predictions

In the second experiment, we investigated which aspects of PROMODES-H in comparison to PROMODES led to the above described differences in performance. For this reason we broke down the summary measures of precision and recall into their original components: true/false positive (TP/FP) and negative (TN/FN) counts presented in the 2×2 contingency table of Figure 1. For general evidence, we averaged across all experiments using relative frequencies. Note that the relative frequencies of positives (TP + FN) and negatives (TN + FP) each sum to one.

The goal was to find out how predictions in each word position changed when applying PROMODES-H instead of PROMODES. This would show where the algorithms agree and where they disagree. PROMODES classifies non-boundaries in 0.9472 of the times correctly as TN and in 0.0528 of the times falsely as boundaries (FP). The algorithm correctly labels 0.3045 of the positions as boundaries (TP) and 0.6955 falsely as non-boundaries (FN). We can see that PROMODES follows a rather conservative approach.

When applying PROMODES-H, the majority of the FP's are turned into non-boundaries, however, a slightly higher number of previously correctly labelled non-boundaries are turned into false boundaries. The net change is a 0.0486 increase in FP's which is the reason for the decrease in precision. On the other side, more false non-

boundaries (FN) are turned into boundaries than in the opposite direction with a net increase of 0.0819 of correct boundaries which led to the increased recall. Since the deduction of precision is less than the increase of recall, a better over-all performance of PROMODES-H is achieved.

In summary, PROMODES predicts more accurately non-boundaries whereas PROMODES-H is better at finding morpheme boundaries. So far we have based our decision for placing a boundary in a certain word position on Equation 2 and 4 assuming that $P(b_{ji}{=}1|\ldots) > P(b_{ji}{=}0|\ldots)$[6] gives the best result. However, if the underlying distribution for boundaries given the evidence is skewed, it might be possible to improve results by introducing a certain decision threshold for inserting morpheme boundaries. We will put this idea to the test in the following section.

3.3 Calibration of the decision threshold

For the third experiment we slightly changed our experimental setup. Instead of dividing datasets during 10-fold cross-validation into training and test subsets with the ratio of 9:1 we randomly split the data into training, validation and test sets with the ratio of 8:1:1. We then run our experiments and measured contingency table counts.

Rather than placing a boundary if $P(b_{ji}{=}1|\ldots) > P(b_{ji}{=}0|\ldots)$ which corresponds to $P(b_{ji}{=}1|\ldots) > 0.50$ we introduced a decision threshold $P(b_{ji}{=}1|\ldots) > h$ with $0 \leq h \leq 1$. This is based on the assumption that the underlying distribution $P(b_{ji}|\ldots)$ might be skewed and an optimal decision can be achieved at a different threshold. The optimal threshold was sought on the validation set and evaluated on the test set. An overview over the validation and test results is given in Table 2. We want to point out that the threshold which yields the best f-measure result on the validation set returns almost the same result on the separate test set for both algorithms which suggests the existence of a general optimal threshold.

Since this experiment provided us with a set of data points where the recall varied monotonically with the threshold and the precision changed accordingly, we reverted to *precision-recall curves* (PR curves) from machine learning. Following Davis and Goadrich (2006) the algorithmic performance can be analysed more informatively using these kinds of curves. The PR curve is plotted with recall on the x-axis and precision on the y-axis for increasing thresholds h. The PR curves for PROMODES and PROMODES-H are shown in Figure 2 on the validation set from which we learnt our optimal thresholds h^*. Points were connected for readability only – points on the PR curve cannot be interpolated linearly.

In addition to the PR curves, we plotted isometrics for corresponding f-measure values which are defined as $precision = \frac{f\text{-}measure \cdot recall}{2recall - f\text{-}measure}$ and are hyperboles. For increasing f-measure values the isometrics are moving further to the top-right corner of the plot. For a threshold of $h = 0.50$ (marked by '$\diamond$') PROMODES-H has a better performance than PROMODES. Nevertheless, across the entire PR curve none of the algorithms dominates. One curve would *dominate* another if all data points of the dominated curve were beneath or equal to the dominating one. PROMODES has its optimal threshold at $h^* = 0.36$ and PROMODES-H at $h^* = 0.37$ where PROMODES has a slightly higher f-measure than PROMODES-H. The points of optimal f-measure performance are marked with '$\triangle$' on the PR curve.

	Prec.	**Recall**	**F-meas.**
PROMODES validation (h=0.50)	0.7522	0.3087	0.4378
PROMODES test (h=0.50)	0.7540	0.3084	0.4378
PROMODES validation (h^*=0.36)	0.5857	0.7824	0.6699
PROMODES test (h^*=0.36)	0.5869	0.7803	0.6699
PROMODES-H validation (h=0.50)	0.6983	0.5333	0.6047
PROMODES-H test (h=0.50)	0.6960	0.5319	0.6030
PROMODES-H validation (h^*=0.37)	0.5848	0.7491	0.6568
PROMODES-H test (h^*=0.37)	0.5857	0.7491	0.6574

Table 2: PROMODES and PROMODES-H on validation and test set.

Summarizing, we have shown that both algorithms commit different errors at the word position level whereas PROMODES is better in predicting non-boundaries and PROMODES-H gives better results for morpheme boundaries at the default threshold of $h = 0.50$. In this section, we demonstrated that across different decision thresholds h for $P(b_{ji}{=}1|\ldots) > h$ none of algorithms dominates the other one, and at the optimal threshold PROMODES achieves a slightly higher performance than PROMODES-H. The question which arises is whether we can combine PROMODES and PROMODES-H in an *ensemble* that leverages individual strengths of both.

[6] Based on Equation 2 and 4 we use the notation $P(b_{ji}|\ldots)$ if we do not want to specify the algorithm.

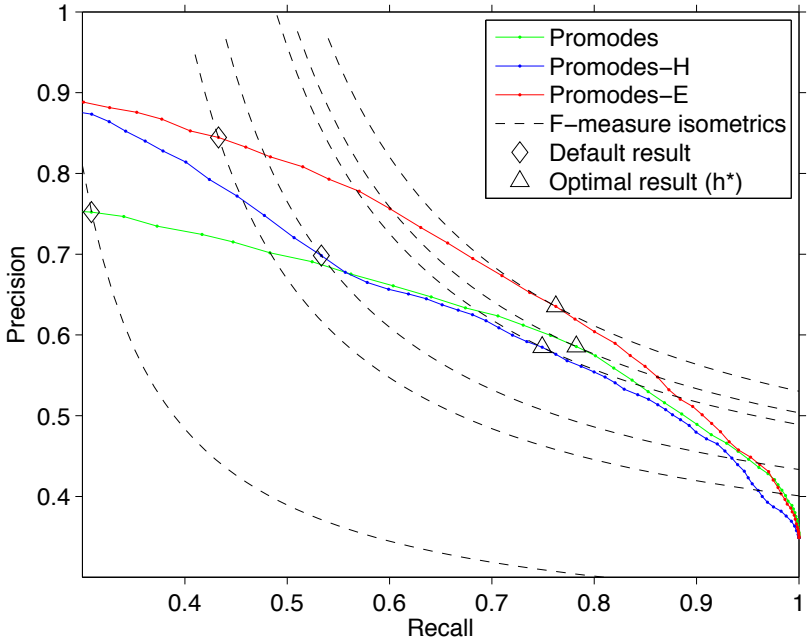


Figure 2: Precision-recall curves for algorithms on validation set.

3.4 A model ensemble to leverage individual strengths

A *model ensemble* is a set of individually trained classifiers whose predictions are combined when classifying new instances (Opitz and Maclin, 1999). The idea is that by combining PROMODES and PROMODES-H, we would be able to avoid certain errors each model commits by consulting the other model as well. We introduce PROMODES-E as the *ensemble* of PROMODES and PROMODES-H. PROMODES-E accesses the individual probabilities $Pr(b_{ji}{=}1|\dots)$ and simply averages them:

$$\frac{Pr(b_{ji}{=}1|t_{ji}) + Pr(b_{ji}{=}1|t_{ji}, b_{j,i\text{-}1}, t_{j,i\text{-}1})}{2} > h \quad .$$

As before, we used the default threshold $h = 0.50$ and found the calibrated threshold $h^* = 0.38$, marked with '◇' and '△' in Figure 2 and shown in Table 3. The calibrated threshold improves the f-measure over both PROMODES and PROMODES-H.

	Prec.	Recall	F-meas.
PROMODES-E validation (h=0.50)	0.8445	0.4328	0.5723
PROMODES-E test (h=0.50)	0.8438	0.4352	0.5742
PROMODES-E validation (h^*=0.38)	0.6354	0.7625	0.6931
PROMODES-E test (h^*=0.38)	0.6350	0.7620	0.6927

Table 3: PROMODES-E on validation and test set.

The optimal solution applying $h^* = 0.38$ is more balanced between precision and recall and boosted the original result by 0.1185 on the test set. Compared to its components PROMODES and PROMODES-H the f-measure increased by 0.0228 and 0.0353 on the test set.

In short, we have shown that by combining PROMODES and PROMODES-H and finding the optimal threshold, the ensemble PROMODES-E gives better results than the individual models themselves and therefore manages to leverage the individual strengths of both to a certain extend. However, can we pinpoint the exact contribution of each individual algorithm to the improved result? We try to find an answer to this question in the analysis of the subsequent section.

3.5 Analysis of calibrated algorithms and their model ensemble

For the entire dataset of 2500 words, we have examined boundary predictions dependent on the relative word position. In Figure 3 and 4 we have plotted the absolute counts of correct boundaries (TP) and non-boundaries (TN) which PROMODES predicted but not PROMODES-H, and vice versa, as continuous lines. We furthermore provided the number of individual predictions which were ultimately adopted by PROMODES-E in the ensemble as dashed lines.

In Figure 3a we can see for the default threshold that PROMODES performs better in predicting non-boundaries in the *middle* and the *end* of the word in comparison to PROMODES-H. Figure 3b

shows the statistics for correctly predicted boundaries. Here, PROMODES-H outperforms PROMODES in predicting correct boundaries across the *entire* word length. After the calibration, shown in Figure 4a, PROMODES-H improves the correct prediction of non-boundaries at the *beginning* of the word whereas PROMODES performs better at the *end*. For the boundary prediction in Figure 4b the signal disappears after calibration.

Concluding, it appears that our test language Zulu has certain features which are modelled best with either a lower or higher-order model. Therefore, the ensemble leveraged strengths of both algorithms which led to a better overall performance with a calibrated threshold.

4 Related work

We have presented two probabilistic generative models for word decomposition, PROMODES and PROMODES-H. Another generative model for morphological analysis has been described by Snover and Brent (2001) and Snover et al. (2002), however, they were interested in finding *paradigms* as sets of mutual exclusive operations on a word form whereas we are describing a generative process using morpheme boundaries and resulting letter transitions.

Moreover, our probabilistic models seem to resemble *Hidden Markov Models* (HMMs) by having certain states and transitions. The main difference is that we have dependencies between states as well as between emissions whereas in HMMs emissions only depend on the underlying state.

Combining different morphological analysers has been performed, for example, by Atwell and Roberts (2006) and Spiegler et al. (2009). Their approaches, though, used *majority vote* to decide whether a morpheme boundary is inserted in a certain word position or not. The algorithms themselves were treated as *black-boxes*.

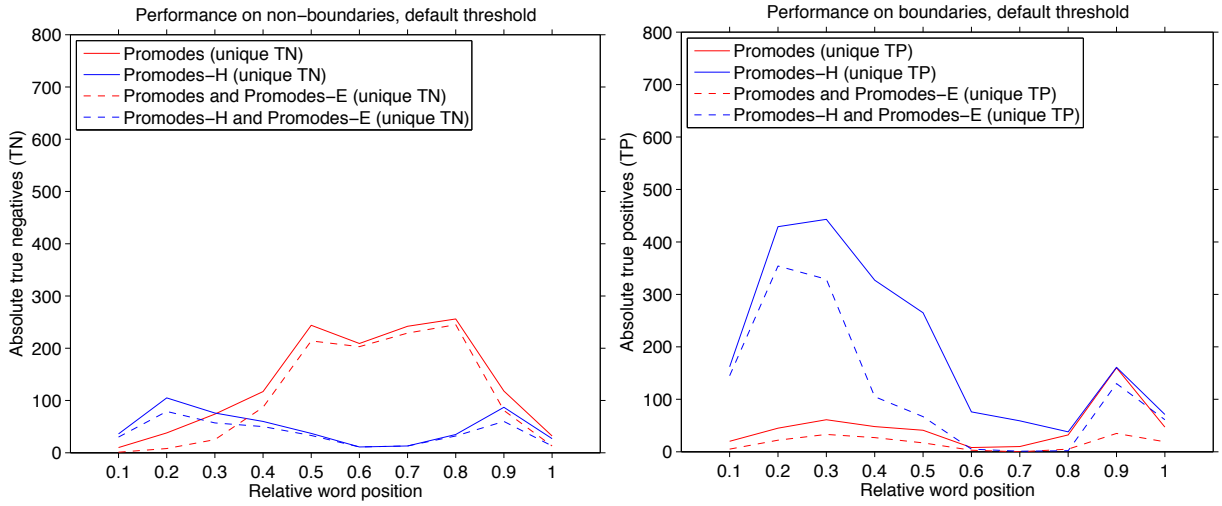
Monson et al. (2009) described an indirect approach to probabilistically combine ParaMor (Monson, 2008) and Morfessor (Creutz, 2006). They used a natural language tagger which was trained on the output of ParaMor and Morfessor. The goal was to mimic each algorithm since ParaMor is rule-based and there is no access to Morfessor's internally used probabilities. The tagger would then return a probability for starting a new morpheme in a certain position based on the original algorithm. These probabilities in combination with a threshold, learnt on a different dataset, were used to merge word analyses. In contrast, our *ensemble* algorithm PROMODES-E directly accesses the probabilistic framework of each algorithm and combines them based on an optimal threshold learnt on a validation set.

5 Conclusions

We have presented a method to learn a *calibrated decision threshold* from a validation set and demonstrated that *ensemble methods* in connection with calibrated decision thresholds can give better results than the individual models themselves. We introduced two algorithms for word decomposition which are based on generative probabilistic models. The models consider segment boundaries as hidden variables and include probabilities for letter transitions within segments. PROMODES contains a lower order model whereas PROMODES-H is a novel development of PROMODES with a higher order model. For both algorithms, we defined the mathematical model and performed experiments on language data of the morphologically complex language *Zulu*. We compared the performance on increasing training set sizes and analysed for each word position whether their boundary prediction agreed or disagreed. We found out that PROMODES was better in predicting non-boundaries and PROMODES-H gave better results for morpheme boundaries at a default decision threshold. At an optimal decision threshold, however, both yielded a similar f-measure result. We then performed a further analysis based on relative word positions and found out that the calibrated PROMODES-H predicted non-boundaries better for initial word positions whereas the calibrated PROMODES for mid- and final word positions. For boundaries, the calibrated algorithms had a similar behaviour. Subsequently, we showed that a *model ensemble* of both algorithms in conjunction with finding an optimal threshold exceeded the performance of the single algorithms at their individually optimal threshold.

Acknowledgements

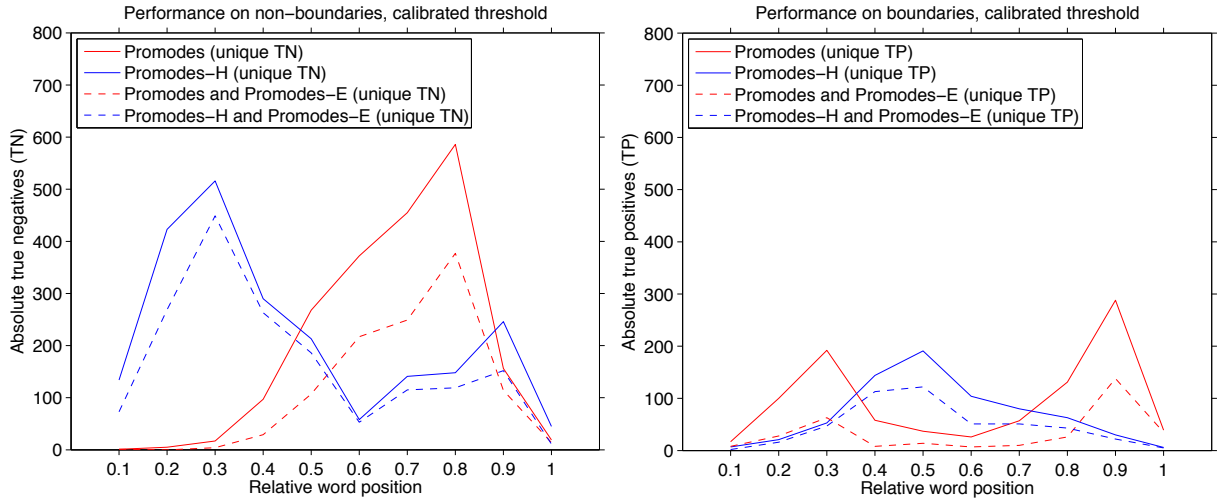
We would like to thank Narayanan Edakunni and Bruno Golénia for discussions concerning this paper as well as the anonymous reviewers for their comments. The research described was sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages*.



(a) True negatives, default

(b) True positives, default

Figure 3: Analysis of results using default threshold.



(a) True negatives, calibrated

(b) True positives, calibrated

Figure 4: Analysis of results using calibrated threshold.

References

J. W. Amtrup. 2003. Morphology in machine translation systems: Efficient integration of finite state transducers and feature structure descriptions. *Machine Translation*, 18(3):217–238.

E. Atwell and A. Roberts. 2006. Combinatory hybrid elementary analysis of text (CHEAT). *Proceedings of the PASCAL Challenges Workshop on Unsupervised Segmentation of Words into Morphemes, Venice, Italy*.

M. Creutz. 2006. *Induction of the Morphology of Natural Language: Unsupervised Morpheme Segmentation with Application to Automatic Speech Recognition*. Ph.D. thesis, Helsinki University of Technology, Espoo, Finland.

J. Davis and M. Goadrich. 2006. The relationship between precision-recall and ROC curves. *International Conference on Machine Learning, Pittsburgh, PA*, 233–240.

M. Gasser. 1994. Modularity in a connectionist model of morphology acquisition. *Proceedings of the 15th conference on Computational linguistics*, 1:214–220.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

J. Goldsmith. 2009. *The Handbook of Computational Linguistics, chapter Segmentation and morphology*. Blackwell.

M. A. Hafer and S. F. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385.

Z. S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

T. Hirsimaki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkonen. 2006. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech And Language*, 20(4):515–541.

K. Kettunen. 2009. Reductive and generative approaches to management of morphological variation of keywords in monolingual information retrieval: An overview. *Journal of Documentation*, 65:267 – 290.

M. Kurimo, S. Virpioja, and V. T. Turunen. 2009. Overview and results of Morpho Challenge 2009. *Working notes for the CLEF 2009 Workshop, Corfu, Greece*.

C. Monson, K. Hollingshead, and B. Roark. 2009. Probabilistic ParaMor. *Working notes for the CLEF 2009 Workshop, Corfu, Greece*.

C. Monson. 2008. *ParaMor: From Paradigm Structure To Natural Language Morphology Induction*. Ph.D. thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

R. J. Mooney and M. E. Califf. 1996. Learning the past tense of English verbs using inductive logic programming. *Symbolic, Connectionist, and Statistical Approaches to Learning for Natural Language Processing*, 370–384.

S. Muggleton and M. Bain. 1999. Analogical prediction. *Inductive Logic Programming: 9th International Workshop, ILP-99, Bled, Slovenia*, 234.

K. Oflazer, S. Nirenburg, and M. McShane. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational. Linguistics*, 27(1):59–85.

D. Opitz and R. Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

D. E. Rumelhart and J. L. McClelland. 1986. *On learning the past tenses of English verbs*. MIT Press, Cambridge, MA, USA.

K. Shalonova, B. Golénia, and P. A. Flach. 2009. Towards learning morphology for under-resourced fusional and agglutinating languages. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):956965.

M. G. Snover and M. R. Brent. 2001. A Bayesian model for morpheme and paradigm identification. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, 490 – 498.

M. G. Snover, G. E. Jarosz, and M. R. Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, 6:11–20.

S. Spiegler, B. Golénia, K. Shalonova, P. A. Flach, and R. Tucker. 2008. Learning the morphology of Zulu with different degrees of supervision. *IEEE Workshop on Spoken Language Technology*.

S. Spiegler, B. Golénia, and P. A. Flach. 2009. Promodes: A probabilistic generative model for word decomposition. *Working Notes for the CLEF 2009 Workshop, Corfu, Greece*.

S. Spiegler, B. Golénia, and P. A. Flach. 2010a. Unsupervised word decomposition with the Promodes algorithm. *In Multilingual Information Access Evaluation Vol. I, CLEF 2009, Corfu, Greece, Lecture Notes in Computer Science, Springer*.

S. Spiegler, A. v. d. Spuy, and P. A. Flach. 2010b. Ukwabelana - An open-source morphological Zulu corpus. *in review*.

R. Sproat. 1996. Multilingual text analysis for text-to-speech synthesis. *Nat. Lang. Eng.*, 2(4):369–380.

Word representations: A simple and general method for semi-supervised learning

Joseph Turian
Département d'Informatique et Recherche Opérationnelle (DIRO)
Université de Montréal
Montréal, Québec, Canada, H3T 1J4
lastname@iro.umontreal.ca

Lev Ratinov
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
ratinov2@uiuc.edu

Yoshua Bengio
Département d'Informatique et Recherche Opérationnelle (DIRO)
Université de Montréal
Montréal, Québec, Canada, H3T 1J4
bengioy@iro.umontreal.ca

Abstract

If we take an existing supervised NLP system, a simple and general way to improve accuracy is to use unsupervised word representations as extra word features. We evaluate Brown clusters, Collobert and Weston (2008) embeddings, and HLBL (Mnih & Hinton, 2009) embeddings of words on both NER and chunking. We use near state-of-the-art supervised baselines, and find that each of the three word representations improves the accuracy of these baselines. We find further improvements by combining different word representations. You can download our word features, for off-the-shelf use in existing NLP systems, as well as our code, here: http://metaoptimize.com/projects/wordreprs/

1 Introduction

By using unlabelled data to reduce data sparsity in the labeled training data, semi-supervised approaches improve generalization accuracy. Semi-supervised models such as Ando and Zhang (2005), Suzuki and Isozaki (2008), and Suzuki et al. (2009) achieve state-of-the-art accuracy. However, these approaches dictate a particular choice of model and training regime. It can be tricky and time-consuming to adapt an existing supervised NLP system to use these semi-supervised techniques. It is preferable to use a simple and general method to adapt existing supervised NLP systems to be semi-supervised.

One approach that is becoming popular is to use unsupervised methods to induce word features—or to download word features that have already been induced—plug these word features into an existing system, and observe a significant increase in accuracy. But which word features are good for what tasks? Should we prefer certain word features? Can we combine them?

A *word representation* is a mathematical object associated with each word, often a vector. Each dimension's value corresponds to a feature and might even have a semantic or grammatical interpretation, so we call it a *word feature*. Conventionally, supervised lexicalized NLP approaches take a word and convert it to a symbolic ID, which is then transformed into a feature vector using a *one-hot* representation: The feature vector has the same length as the size of the vocabulary, and only one dimension is on. However, the one-hot representation of a word suffers from data sparsity: Namely, for words that are rare in the labeled training data, their corresponding model parameters will be poorly estimated. Moreover, at test time, the model cannot handle words that do not appear in the labeled training data. These limitations of one-hot word representations have prompted researchers to investigate unsupervised methods for inducing word representations over large unlabeled corpora. Word features can be hand-designed, but our goal is to learn them.

One common approach to inducing unsupervised word representation is to use clustering, perhaps hierarchical. This technique was used by a variety of researchers (Miller et al., 2004; Liang, 2005; Koo et al., 2008; Ratinov & Roth, 2009; Huang & Yates, 2009). This leads to a one-hot representation over a smaller vocabulary size. Neural language models (Bengio et al., 2001; Schwenk & Gauvain, 2002; Mnih & Hinton, 2007; Collobert & Weston, 2008), on the other hand, induce dense real-valued low-dimensional

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 384–394,
Uppsala, Sweden, 11-16 July 2010.

word embeddings using unsupervised approaches. (See Bengio (2008) for a more complete list of references on neural language models.)

Unsupervised word representations have been used in previous NLP work, and have demonstrated improvements in generalization accuracy on a variety of tasks. But different word representations have never been systematically compared in a controlled way. In this work, we compare different techniques for inducing word representations, evaluating them on the tasks of named entity recognition (NER) and chunking.

We retract former negative results published in Turian et al. (2009) about Collobert and Weston (2008) embeddings, given training improvements that we describe in Section 7.1.

2 Distributional representations

Distributional word representations are based upon a cooccurrence matrix F of size $W \times C$, where W is the vocabulary size, each row F_w is the initial representation of word w, and each column F_c is some context. Sahlgren (2006) and Turney and Pantel (2010) describe a handful of possible design decisions in contructing F, including choice of context types (left window? right window? size of window?) and type of frequency count (raw? binary? tf-idf?). F_w has dimensionality W, which can be too large to use F_w as features for word w in a supervised model. One can map F to matrix f of size $W \times d$, where $d \ll C$, using some function g, where $f = g(F)$. f_w represents word w as a vector with d dimensions. The choice of g is another design decision, although perhaps not as important as the statistics used to initially construct F.

The self-organizing semantic map (Ritter & Kohonen, 1989) is a distributional technique that maps words to two dimensions, such that syntactically and semantically related words are nearby (Honkela et al., 1995; Honkela, 1997).

LSA (Dumais et al., 1988; Landauer et al., 1998), LSI, and LDA (Blei et al., 2003) induce distributional representations over F in which each column is a *document* context. In most of the other approaches discussed, the columns represent word contexts. In LSA, g computes the SVD of F.

Hyperspace Analogue to Language (HAL) is another early distributional approach (Lund et al., 1995; Lund & Burgess, 1996) to inducing word representations. They compute F over a corpus of 160 million word tokens with a vocabulary size W of 70K word types. There are $2 \cdot W$ types of context (columns): The first or second W are counted if the word c occurs within a window of 10 to the left or right of the word w, respectively. f is chosen by taking the 200 columns (out of 140K in F) with the highest variances. ICA is another technique to transform F into f. (Väyrynen & Honkela, 2004; Väyrynen & Honkela, 2005; Väyrynen et al., 2007). ICA is expensive, and the largest vocabulary size used in these works was only 10K. As far as we know, ICA methods have not been used when the size of the vocab W is 100K or more.

Explicitly storing cooccurrence matrix F can be memory-intensive, and transforming F to f can be time-consuming. It is preferable that F never be computed explicitly, and that f be constructed *incrementally*. Řehůřek and Sojka (2010) describe an incremental approach to inducing LSA and LDA topic models over 270 millions word tokens with a vocabulary of 315K word types. This is similar in magnitude to our experiments.

Another incremental approach to constructing f is using a random projection: Linear mapping g is multiplying F by a random matrix chosen *a priori*. This *random indexing* method is motivated by the Johnson-Lindenstrauss lemma, which states that for certain choices of random matrix, if d is sufficiently large, then the original distances between words in F will be preserved in f (Sahlgren, 2005). Kaski (1998) uses this technique to produce 100-dimensional representations of documents. Sahlgren (2001) was the first author to use random indexing using narrow context. Sahlgren (2006) does a battery of experiments exploring different design decisions involved in constructing F, prior to using random indexing. However, like all the works cited above, Sahlgren (2006) only uses distributional representation to improve existing systems for one-shot classification tasks, such as IR, WSD, semantic knowledge tests, and text categorization. It is not well-understood what settings are appropriate to induce distributional word representations for structured prediction tasks (like parsing and MT) and sequence labeling tasks (like chunking and NER). Previous research has achieved repeated successes on these tasks using clustering representations (Section 3) and distributed representations (Section 4), so we focus on these representations in our work.

3 Clustering-based word representations

Another type of word representation is to induce a clustering over words. Clustering methods and

distributional methods can overlap. For example, Pereira et al. (1993) begin with a cooccurrence matrix and transform this matrix into a clustering.

3.1 Brown clustering

The Brown algorithm is a hierarchical clustering algorithm which clusters words to maximize the mutual information of bigrams (Brown et al., 1992). So it is a class-based bigram language model. It runs in time $O(V \cdot K^2)$, where V is the size of the vocabulary and K is the number of clusters.

The hierarchical nature of the clustering means that we can choose the word class at several levels in the hierarchy, which can compensate for poor clusters of a small number of words. One downside of Brown clustering is that it is based solely on bigram statistics, and does not consider word usage in a wider context.

Brown clusters have been used successfully in a variety of NLP applications: NER (Miller et al., 2004; Liang, 2005; Ratinov & Roth, 2009), PCFG parsing (Candito & Crabbé, 2009), dependency parsing (Koo et al., 2008; Suzuki et al., 2009), and semantic dependency parsing (Zhao et al., 2009).

Martin et al. (1998) presents algorithms for inducing hierarchical clusterings based upon word bigram *and trigram* statistics. Ushioda (1996) presents an extension to the Brown clustering algorithm, and learn hierarchical clusterings of words *as well as phrases*, which they apply to POS tagging.

3.2 Other work on cluster-based word representations

Lin and Wu (2009) present a K-means-like non-hierarchical clustering algorithm for phrases, which uses MapReduce.

HMMs can be used to induce a soft clustering, specifically a multinomial distribution over possible clusters (hidden states). Li and McCallum (2005) use an HMM-LDA model to improve POS tagging and Chinese Word Segmentation. Huang and Yates (2009) induce a fully-connected HMM, which emits a multinomial distribution over possible vocabulary words. They perform hard clustering using the Viterbi algorithm. (Alternately, they could keep the soft clustering, with the representation for a particular word token being the posterior probability distribution over the states.) However, the CRF chunker in Huang and Yates (2009), which uses their HMM word clusters as extra features, achieves F1 lower than a baseline CRF chunker (Sha & Pereira, 2003). Goldberg et al. (2009) use an HMM to assign POS tags to words, which in turns improves the accuracy of the PCFG-based Hebrew parser. Deschacht and Moens (2009) use a latent-variable language model to improve semantic role labeling.

4 Distributed representations

Another approach to word representation is to learn a distributed representation. (Not to be confused with *distributional* representations.) A distributed representation is dense, low-dimensional, and real-valued. Distributed word representations are called *word embeddings*. Each dimension of the embedding represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties. A distributed representation is compact, in the sense that it can represent an exponential number of clusters in the number of dimensions.

Word embeddings are typically induced using *neural language models*, which use neural networks as the underlying predictive model (Bengio, 2008). Historically, training and testing of neural language models has been slow, scaling as the size of the vocabulary for each model computation (Bengio et al., 2001; Bengio et al., 2003). However, many approaches have been proposed in recent years to eliminate that linear dependency on vocabulary size (Morin & Bengio, 2005; Collobert & Weston, 2008; Mnih & Hinton, 2009) and allow scaling to very large training corpora.

4.1 Collobert and Weston (2008) embeddings

Collobert and Weston (2008) presented a neural language model that could be trained over billions of words, because the gradient of the loss was computed stochastically over a small sample of possible outputs, in a spirit similar to Bengio and Sénécal (2003). This neural model of Collobert and Weston (2008) was refined and presented in greater depth in Bengio et al. (2009).

The model is discriminative and non-probabilistic. For each training update, we read an n-gram $x = (w_1, \ldots, w_n)$ from the corpus. The model concatenates the learned embeddings of the n words, giving $e(w_1) \oplus \ldots \oplus e(w_n)$, where e is the lookup table and $\oplus$ is concatenation. We also create a *corrupted* or *noise* n-gram $\tilde{x} = (w_1, \ldots, w_{n-q}, \tilde{w_n})$, where $\tilde{w_n} \neq w_n$ is chosen uniformly from the vocabulary.[1] For convenience,

[1] In Collobert and Weston (2008), the middle word in the

we write $e(x)$ to mean $e(w_1) \oplus \ldots \oplus e(w_n)$. We predict a score $s(x)$ for x by passing $e(x)$ through a single hidden layer neural network. The training criterion is that n-grams that are present in the training corpus like x must have a score at least some margin higher than corrupted n-grams like $\tilde{x}$. Specifically: $L(x) = \max(0, 1 - s(x) + s(\tilde{x}))$. We minimize this loss stochastically over the n-grams in the corpus, doing gradient descent simultaneously over the neural network parameters *and* the embedding lookup table.

We implemented the approach of Collobert and Weston (2008), with the following differences:

- We did not achieve as low log-ranks on the English Wikipedia as the authors reported in Bengio et al. (2009), despite initially attempting to have identical experimental conditions.
- We corrupt the *last* word of each n-gram.
- We had a separate learning rate for the embeddings and for the neural network weights. We found that the embeddings should have a learning rate generally 1000–32000 times higher than the neural network weights. Otherwise, the unsupervised training criterion drops slowly.
- Although their sampling technique makes training fast, testing is still expensive when the size of the vocabulary is large. Instead of cross-validating using the log-rank over the validation data as they do, we instead used the moving average of the training loss on training examples before the weight update.

4.2 HLBL embeddings

The log-bilinear model (Mnih & Hinton, 2007) is a probabilistic and linear neural model. Given an n-gram, the model concatenates the embeddings of the $n - 1$ first words, and learns a linear model to predict the embedding of the last word. The similarity between the predicted embedding and the current actual embedding is transformed into a probability by exponentiating and then normalizing. Mnih and Hinton (2009) speed up model evaluation during training and testing by using a hierarchy to exponentially filter down the number of computations that are performed. This hierarchical evaluation technique was first proposed by Morin and Bengio (2005). The model, combined with this optimization, is called the *hierarchical log-bilinear (HLBL)* model.

n-gram is corrupted. In Bengio et al. (2009), the last word in the n-gram is corrupted.

5 Supervised evaluation tasks

We evaluate the hypothesis that one can take an existing, near state-of-the-art, supervised NLP system, and improve its accuracy by including word representations as word features. This technique for turning a supervised approach into a semi-supervised one is general and task-agnostic.

However, we wish to find out if certain word representations are preferable for certain tasks. Lin and Wu (2009) finds that the representations that are good for NER are poor for search query classification, and vice-versa. We apply clustering and distributed representations to NER and chunking, which allows us to compare our semi-supervised models to those of Ando and Zhang (2005) and Suzuki and Isozaki (2008).

5.1 Chunking

Chunking is a syntactic sequence labeling task. We follow the conditions in the CoNLL-2000 shared task (Sang & Buchholz, 2000).

The linear CRF chunker of Sha and Pereira (2003) is a standard near-state-of-the-art baseline chunker. In fact, many off-the-shelf CRF implementations now replicate Sha and Pereira (2003), including their choice of feature set:

- CRF++ by Taku Kudo (`http://crfpp.sourceforge.net/`)
- crfsgd by Léon Bottou (`http://leon.bottou.org/projects/sgd`)
- CRFsuite by by Naoaki Okazaki (`http://www.chokkan.org/software/crfsuite/`)

We use CRFsuite because it makes it simple to modify the feature generation code, so one can easily add new features. We use SGD optimization, and enable negative state features and negative transition features. ("`feature.possible_transitions=1, feature.possible_states=1`")

Table 1 shows the features in the baseline chunker. As you can see, the Brown and embedding features are unigram features, and do not participate in conjunctions like the word features and tag features do. Koo et al. (2008) sees further accuracy improvements on dependency parsing when using word representations in compound features.

The data comes from the Penn Treebank, and is newswire from the Wall Street Journal in 1989. Of the 8936 training sentences, we used 1000 randomly sampled sentences (23615 words) for development. We trained models on the 7936

- Word features: w_i for i in $\{-2,-1,0,+1,+2\}$, $w_i \wedge w_{i+1}$ for i in $\{-1,0\}$.
- Tag features: w_i for i in $\{-2,-1,0,+1,+2\}$, $t_i \wedge t_{i+1}$ for i in $\{-2,-1,0,+1\}$. $t_i \wedge t_{i+1} \wedge t_{i+2}$ for i in $\{-2,-1,0\}$.
- Embedding features [if applicable]: $e_i[d]$ for i in $\{-2,-1,0,+1,+2\}$, where d ranges over the dimensions of the embedding e_i.
- Brown features [if applicable]: $substr(b_i, 0, p)$ for i in $\{-2,-1,0,+1,+2\}$, where *substr* takes the p-length prefix of the Brown cluster b_i.

Table 1: Features templates used in the CRF chunker.

training partition sentences, and evaluated their F1 on the development set. After choosing hyperparameters to maximize the dev F1, we would retrain the model using these hyperparameters on the full 8936 sentence training set, and evaluate on test. One hyperparameter was l2-regularization sigma, which for most models was optimal at 2 or 3.2. The word embeddings also required a scaling hyperparameter, as described in Section 7.2.

5.2 Named entity recognition

NER is typically treated as a sequence prediction problem. Following Ratinov and Roth (2009), we use the regularized averaged perceptron model. Ratinov and Roth (2009) describe different sequence encoding like BILOU and BIO, and show that the BILOU encoding outperforms BIO, and the greedy inference performs competitively to Viterbi while being significantly faster. Accordingly, we use greedy inference and BILOU text chunk representation. We use the publicly available implementation from Ratinov and Roth (2009) (see the end of this paper for the URL). In our baseline experiments, we remove gazetteers and non-local features (Krishnan & Manning, 2006). However, we also run experiments that include these features, to understand if the information they provide mostly overlaps with that of the word representations.

After each epoch over the training set, we measured the accuracy of the model on the development set. Training was stopped after the accuracy on the development set did not improve for 10 epochs, generally about 50–80 epochs total. The epoch that performed best on the development set was chosen as the final model.

We use the following baseline set of features from Zhang and Johnson (2003):
- Previous two predictions y_{i-1} and y_{i-2}
- Current word x_i
- x_i word type information: all-capitalized, is-capitalized, all-digits, alphanumeric, etc.
- Prefixes and suffixes of x_i, if the word contains hyphens, then the tokens between the hyphens
- Tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
- Capitalization pattern in the window c
- Conjunction of c and y_{i-1}.

Word representation features, if present, are used the same way as in Table 1.

When using the lexical features, we normalize dates and numbers. For example, *1980* becomes **DDDD** and *212-325-4751* becomes **DDD*-*DDD*-*DDDD**. This allows a degree of abstraction to years, phone numbers, etc. This delexicalization is performed separately from using the word representation. That is, if we have induced an embedding for *12/3/2008* , we will use the embedding of *12/3/2008* , and **DD*/*D*/*DDDD** in the baseline features listed above.

Unlike in our chunking experiments, after we chose the best model on the development set, we used that model on the test set too. (In chunking, after finding the best hyperparameters on the development set, we would combine the dev and training set and training a model over this combined set, and then evaluate on test.)

The standard evaluation benchmark for NER is the CoNLL03 shared task dataset drawn from the Reuters newswire. The training set contains 204K words (14K sentences, 946 documents), the test set contains 46K words (3.5K sentences, 231 documents), and the development set contains 51K words (3.3K sentences, 216 documents).

We also evaluated on an out-of-domain (OOD) dataset, the MUC7 formal run (59K words). MUC7 has a different annotation standard than the CoNLL03 data. It has several NE types that don't appear in CoNLL03: money, dates, and numeric quantities. CoNLL03 has MISC, which is not present in MUC7. To evaluate on MUC7, we perform the following postprocessing steps prior to evaluation:

1. In the gold-standard MUC7 data, discard (label as 'O') all NEs with type NUMBER/MONEY/DATE.
2. In the predicted model output on MUC7 data, discard (label as 'O') all NEs with type MISC.

These postprocessing steps will adversely affect all NER models across-the-board, nonetheless allowing us to compare different models in a controlled manner.

6 Unlabled Data

Unlabeled data is used for inducing the word representations. We used the RCV1 corpus, which contains one year of Reuters English newswire, from August 1996 to August 1997, about 63 millions words in 3.3 million sentences. We left case intact in the corpus. By comparison, Collobert and Weston (2008) downcases words and delexicalizes numbers.

We use a preprocessing technique proposed by Liang, (2005, p. 51), which was later used by Koo et al. (2008): Remove all sentences that are less than 90% lowercase a–z. We assume that whitespace is not counted, although this is not specified in Liang's thesis. We call this preprocessing step *cleaning*.

In Turian et al. (2009), we found that all word representations performed better on the supervised task when they were induced on the clean unlabeled data, both embeddings and Brown clusters. This is the case even though the cleaning process was very aggressive, and discarded more than half of the sentences. According to the evidence and arguments presented in Bengio et al. (2009), the non-convex optimization process for Collobert and Weston (2008) embeddings might be adversely affected by noise and the statistical sparsity issues regarding rare words, especially at the beginning of training. For this reason, we hypothesize that learning representations over the most frequent words first and gradually increasing the vocabulary—a *curriculum* training strategy (Elman, 1993; Bengio et al., 2009; Spitkovsky et al., 2010)—would provide better results than cleaning.

After cleaning, there are 37 million words (58% of the original) in 1.3 million sentences (41% of the original). The cleaned RCV1 corpus has 269K word types. This is the vocabulary size, i.e. how many word representations were induced. Note that cleaning is applied only to the unlabeled data, not to the labeled data used in the supervised tasks.

RCV1 is a superset of the CoNLL03 corpus. For this reason, NER results that use RCV1 word representations are a form of transductive learning.

7 Experiments and Results

7.1 Details of inducing word representations

The Brown clusters took roughly 3 days to induce, when we induced 1000 clusters, the baseline in prior work (Koo et al., 2008; Ratinov & Roth, 2009). We also induced 100, 320, and 3200 Brown clusters, for comparison. (Because Brown clustering scales quadratically in the number of clusters, inducing 10000 clusters would have been prohibitive.) Because Brown clusters are hierarchical, we can use cluster supersets as features. We used clusters at path depth 4, 6, 10, and 20 (Ratinov & Roth, 2009). These are the prefixes used in Table 1.

The Collobert and Weston (2008) (C&W) embeddings were induced over the course of a few weeks, and trained for about 50 epochs. One of the difficulties in inducing these embeddings is that there is no stopping criterion defined, and that the quality of the embeddings can keep improving as training continues. Collobert (p.c.) simply leaves one computer training his embeddings indefinitely. We induced embeddings with 25, 50, 100, or 200 dimensions over 5-gram windows. In comparison to Turian et al. (2009), we use improved C&W embeddings in this work:

• They were trained for 50 epochs, not just 20 epochs.

• We initialized all embedding dimensions uniformly in the range [-0.01, +0.01], not [-1,+1]. For rare words, which are typically updated only 143 times per epoch[2], and given that our embedding learning rate was typically 1e-6 or 1e-7, this means that rare word embeddings will be concentrated around zero, instead of spread out randomly.

The HLBL embeddings were trained for 100 epochs (7 days).[3] Unlike our Collobert and Weston (2008) embeddings, we did not extensively tune the learning rates for HLBL. We used a learning rate of 1e-3 for both model parameters and embedding parameters. We induced embeddings with 100 dimensions over 5-gram windows, and embeddings with 50 dimensions over 5-gram windows. Embeddings were induced over one pass

[2] A rare word will appear 5 (window size) times per epoch as a positive example, and 37M (training examples per epoch) / 269K (vocabulary size) = 138 times per epoch as a corruption example.

[3] The HLBL model updates require fewer matrix multiplies than Collobert and Weston (2008) model updates. Additionally, HLBL models were trained on a GPGPU, which is faster than conventional CPU arithmetic.

approach using a random tree, not two passes with an updated tree and embeddings re-estimation.

7.2 Scaling of Word Embeddings

Like many NLP systems, the baseline system contains only binary features. The word embeddings, however, are real numbers that are not necessarily in a bounded range. If the range of the word embeddings is too large, they will exert more influence than the binary features.

We generally found that embeddings had zero mean. We can scale the embeddings by a hyperparameter, to control their standard deviation. Assume that the embeddings are represented by a matrix E:

$$E \leftarrow \sigma \cdot E / stddev(E) \quad (1)$$

σ is a scaling constant that sets the new standard deviation after scaling the embeddings.

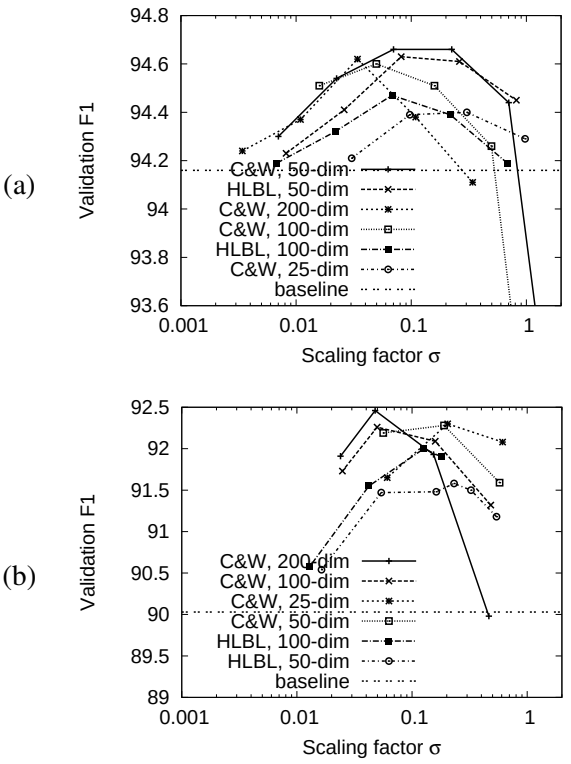


Figure 1: Effect as we vary the scaling factor σ (Equation 1) on the validation set F1. We experiment with Collobert and Weston (2008) and HLBL embeddings of various dimensionality. (a) Chunking results. (b) NER results.

Figure 1 shows the effect of scaling factor σ on both supervised tasks. We were surprised to find that on both tasks, across Collobert and Weston (2008) and HLBL embeddings of various dimensionality, that all curves had similar shapes and optima. This is one contributions of our work. In Turian et al. (2009), we were not able to prescribe a default value for scaling the embeddings. However, these curves demonstrate that a reasonable choice of scale factor is such that the embeddings have a standard deviation of 0.1.

7.3 Capacity of Word Representations

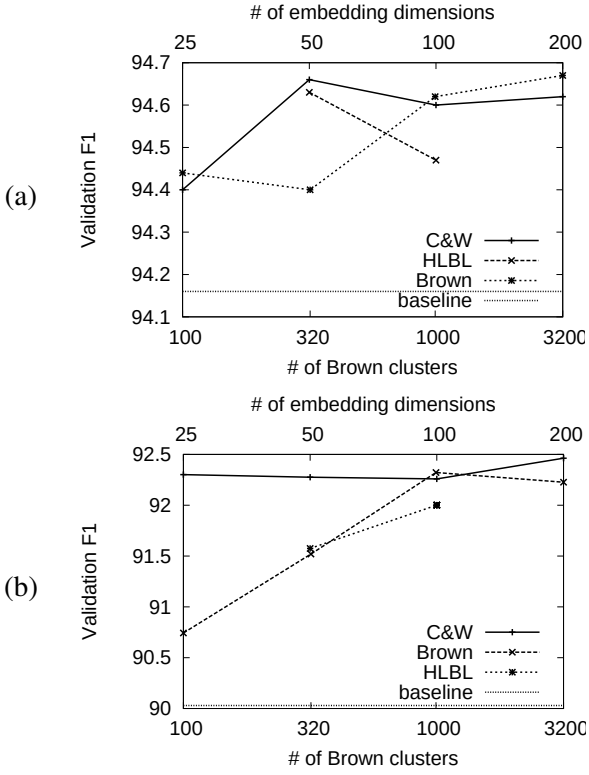


Figure 2: Effect as we vary the capacity of the word representations on the validation set F1. (a) Chunking results. (b) NER results.

There are capacity controls for the word representations: number of Brown clusters, and number of dimensions of the word embeddings. Figure 2 shows the effect on the validation F1 as we vary the capacity of the word representations.

In general, it appears that more Brown clusters are better. We would like to induce 10000 Brown clusters, however this would take several months.

In Turian et al. (2009), we hypothesized on the basis of solely the HLBL NER curve that higher-dimensional word embeddings would give higher accuracy. Figure 2 shows that this hypothesis is not true. For NER, the C&W curve is almost flat, and we were suprised to find the even 25-dimensional C&W word embeddings work so well. For chunking, 50-dimensional embeddings had the highest validation F1 for both C&W and HLBL. These curves indicates that the optimal capacity of the word embeddings is task-specific.

System	Dev	Test
Baseline	94.16	93.79
HLBL, 50-dim	94.63	94.00
C&W, 50-dim	94.66	94.10
Brown, 3200 clusters	**94.67**	**94.11**
Brown+HLBL, 37M	94.62	94.13
C&W+HLBL, 37M	94.68	94.25
Brown+C&W+HLBL, 37M	94.72	94.15
Brown+C&W, 37M	94.76	94.35
Ando and Zhang (2005), 15M	-	94.39
Suzuki and Isozaki (2008), 15M	-	94.67
Suzuki and Isozaki (2008), 1B	-	**95.15**

Table 2: Final chunking F1 results. In the last section, we show how many unlabeled words were used.

System	Dev	Test	MUC7
Baseline	90.03	84.39	67.48
Baseline+Nonlocal	91.91	86.52	71.80
HLBL 100-dim	92.00	88.13	75.25
Gazetteers	92.09	87.36	77.76
C&W 50-dim	92.27	87.93	75.74
Brown, 1000 clusters	92.32	**88.52**	**78.84**
C&W 200-dim	**92.46**	87.96	75.51
C&W+HLBL	92.52	88.56	78.64
Brown+HLBL	92.56	88.93	77.85
Brown+C&W	92.79	89.31	80.13
HLBL+Gaz	92.91	89.35	79.29
C&W+Gaz	92.98	88.88	81.44
Brown+Gaz	**93.25**	**89.41**	**82.71**
Lin and Wu (2009), 3.4B	-	88.44	-
Ando and Zhang (2005), 27M	93.15	89.31	-
Suzuki and Isozaki (2008), 37M	93.66	89.36	-
Suzuki and Isozaki (2008), 1B	**94.48**	89.92	-
All (Brown+C&W+HLBL+Gaz), 37M	93.17	90.04	82.50
All+Nonlocal, 37M	93.95	90.36	84.15
Lin and Wu (2009), 700B	-	**90.90**	-

Table 3: Final NER F1 results, showing the cumulative effect of adding word representations, non-local features, and gazetteers to the baseline. To speed up training, in combined experiments (C&W plus another word representation), we used the 50-dimensional C&W embeddings, not the 200-dimensional ones. In the last section, we show how many unlabeled words were used.

7.4 Final results

Table 2 shows the final chunking results and Table 3 shows the final NER F1 results. We compare to the state-of-the-art methods of Ando and Zhang (2005), Suzuki and Isozaki (2008), and—for NER—Lin and Wu (2009). Tables 2 and 3 show that accuracy can be increased further by combining the features from different types of word representations. But, if only one word representation is to be used, Brown clusters have the highest accuracy. Given the improvements to the C&W embeddings since Turian et al. (2009), C&W embeddings outperform the HLBL embeddings. On chunking, there is only a minute difference between Brown clusters and the embeddings. Combining representations leads to small increases in the test F1. In comparison to chunking, combining different word representations on NER seems gives larger improvements on the test F1.

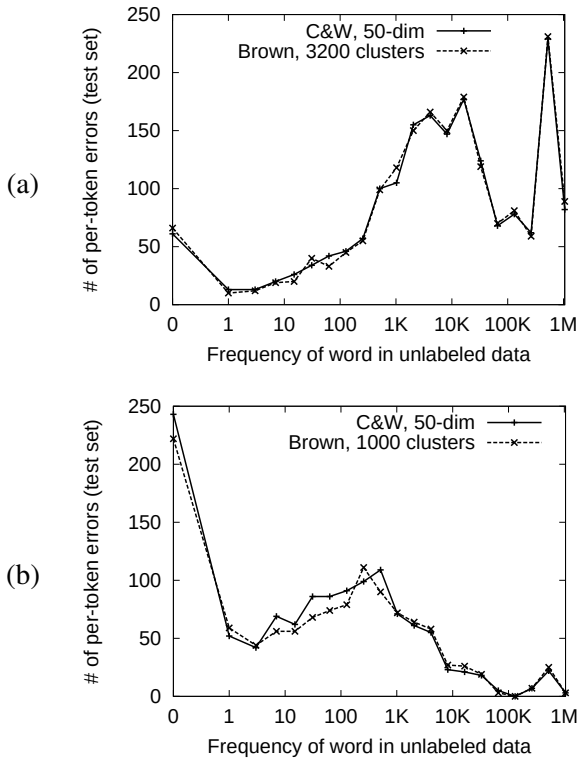


Figure 3: For word tokens that have different frequency in the unlabeled data, what is the total number of per-token errors incurred on the test set? (a) Chunking results. (b) NER results.

On NER, Brown clusters are superior to the word embeddings. Since much of the NER F1 is derived from decisions made over rare words, we suspected that Brown clustering has a superior representation for rare words. Brown makes a single hard clustering decision, whereas the embedding for a rare word is close to its initial value since it hasn't received many training updates (see Footnote 2). Figure 3 shows the total number of per-token errors incurred on the test set, depending upon the frequency of the word token in the unlabeled data. For NER, Figure 3 (b) shows that most errors occur on rare words, and that Brown clusters do indeed incur fewer errors for rare words. This supports our hypothesis that, for rare words, Brown clustering produces better representations than word embeddings that haven't received sufficient training updates. For chunking, Brown clusters and C&W embeddings incur almost identical numbers of errors, and errors are concentrated around the more common

words. We hypothesize that *non-rare* words have good representations, regardless of the choice of word representation technique. For tasks like chunking in which a syntactic decision relies upon looking at several token simultaneously, compound features that use the word representations might increase accuracy more (Koo et al., 2008).

Using word representations in NER brought larger gains on the out-of-domain data than on the in-domain data. We were surprised by this result, because the OOD data was not even used during the unsupervised word representation induction, as was the in-domain data. We are curious to investigate this phenomenon further.

Ando and Zhang (2005) present a semi-supervised learning algorithm called alternating structure optimization (ASO). They find a low-dimensional projection of the input features that gives good linear classifiers over auxiliary tasks. These auxiliary tasks are sometimes specific to the supervised task, and sometimes general language modeling tasks like "predict the missing word". Suzuki and Isozaki (2008) present a semi-supervised extension of CRFs. (In Suzuki et al. (2009), they extend their semi-supervised approach to more general conditional models.) One of the advantages of the semi-supervised learning approach that we use is that it is simpler and more general than that of Ando and Zhang (2005) and Suzuki and Isozaki (2008). Their methods dictate a particular choice of model and training regime and could not, for instance, be used with an NLP system based upon an SVM classifier.

Lin and Wu (2009) present a K-means-like non-hierarchical clustering algorithm for phrases, which uses MapReduce. Since they can scale to millions of phrases, and they train over 800B unlabeled words, they achieve state-of-the-art accuracy on NER using their phrase clusters. This suggests that extending word representations to phrase representations is worth further investigation.

8 Conclusions

Word features can be learned in advance in an unsupervised, task-inspecific, and model-agnostic manner. These word features, once learned, are easily disseminated with other researchers, and easily integrated into existing supervised NLP systems. The disadvantage, however, is that accuracy might not be as high as a semi-supervised method that includes task-specific information and that jointly learns the supervised and unsupervised tasks (Ando & Zhang, 2005; Suzuki & Isozaki, 2008; Suzuki et al., 2009).

Unsupervised word representations have been used in previous NLP work, and have demonstrated improvements in generalization accuracy on a variety of tasks. Ours is the first work to systematically compare different word representations in a controlled way. We found that Brown clusters and word embeddings both can improve the accuracy of a near-state-of-the-art supervised NLP system. We also found that combining different word representations can improve accuracy further. Error analysis indicates that Brown clustering induces better representations for rare words than C&W embeddings that have not received many training updates.

Another contribution of our work is a default method for setting the scaling parameter for word embeddings. With this contribution, word embeddings can now be used off-the-shelf as word features, with no tuning.

Future work should explore methods for inducing phrase representations, as well as techniques for increasing in accuracy by using word representations in *compound* features.

Replicating our experiments

You can visit `http://metaoptimize.com/projects/wordreprs/` to find: The word representations we induced, which you can download and use in your experiments; The code for inducing the word representations, which you can use to induce word representations on your own data; The NER and chunking system, with code for replicating our experiments.

Acknowledgments

Thank you to Magnus Sahlgren, Bob Carpenter, Percy Liang, Alexander Yates, and the anonymous reviewers for useful discussion. Thank you to Andriy Mnih for inducing his embeddings on RCV1 for us. Joseph Turian and Yoshua Bengio acknowledge the following agencies for research funding and computing support: NSERC, RQCHP, CIFAR. Lev Ratinov was supported by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL).

References

Ando, R., & Zhang, T. (2005). A high-performance semi-supervised learning method for text chunking. *ACL*.

Bengio, Y. (2008). Neural net language models. *Scholarpedia*, *3*, 3881.

Bengio, Y., Ducharme, R., & Vincent, P. (2001). A neural probabilistic language model. *NIPS*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *ICML*.

Bengio, Y., & Sénécal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. *AISTATS*.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*, 467–479.

Candito, M., & Crabbé, B. (2009). Improving generative statistical parsing with semi-supervised word clustering. *IWPT* (pp. 138–141).

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *ICML*.

Deschacht, K., & Moens, M.-F. (2009). Semi-supervised semantic role labeling using the Latent Words Language Model. *EMNLP* (pp. 21–29).

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., & Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. *SIGCHI Conference on Human Factors in Computing Systems* (pp. 281–285). ACM.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*, 781–799.

Goldberg, Y., Tsarfaty, R., Adler, M., & Elhadad, M. (2009). Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. *EACL*.

Honkela, T. (1997). Self-organizing maps of words for natural language processing applications. *Proceedings of the International ICSC Symposium on Soft Computing*.

Honkela, T., Pulkki, V., & Kohonen, T. (1995). Contextual relations of words in grimm tales, analyzed by self-organizing map. *ICANN*.

Huang, F., & Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence labeling. *ACL*.

Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering. *IJCNN* (pp. 413–418).

Koo, T., Carreras, X., & Collins, M. (2008). Simple semi-supervised dependency parsing. *ACL* (pp. 595–603).

Krishnan, V., & Manning, C. D. (2006). An effective two-stage model for exploiting non-local dependencies in named entity recognition. *COLING-ACL*.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, 259–284.

Li, W., & McCallum, A. (2005). Semi-supervised sequence modeling with syntactic topic models. *AAAI*.

Liang, P. (2005). Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Lin, D., & Wu, X. (2009). Phrase clustering for discriminative learning. *ACL-IJCNLP* (pp. 1030–1038).

Lund, K., & Burgess, C. (1996). Producing highdimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, *28*, 203–208.

Lund, K., Burgess, C., & Atchley, R. A. (1995). Semantic and associative priming in high-dimensional semantic space. *Cognitive Science Proceedings, LEA* (pp. 660–665).

Martin, S., Liermann, J., & Ney, H. (1998). Algorithms for bigram and trigram word clustering. *Speech Communication*, *24*, 19–37.

Miller, S., Guinness, J., & Zamanian, A. (2004). Name tagging with word clusters and discriminative training. *HLT-NAACL* (pp. 337–342).

Mnih, A., & Hinton, G. E. (2007). Three new graphical models for statistical language modelling. *ICML*.

Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. *NIPS* (pp. 1081–1088).

Morin, F., & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. *AISTATS*.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of english words. *ACL* (pp. 183–190).

Ratinov, L., & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. *CoNLL*.

Ritter, H., & Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 241–254.

Sahlgren, M. (2001). Vector-based semantic analysis: Representing word meanings based on random labels. *Proceedings of the Semantic Knowledge Acquisition and Categorisation Workshop, ESSLLI*.

Sahlgren, M. (2005). An introduction to random indexing. *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*.

Sahlgren, M. (2006). *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Doctoral dissertation, Stockholm University.

Sang, E. T., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *CoNLL*.

Schwenk, H., & Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 765–768). Orlando, Florida.

Sha, F., & Pereira, F. C. N. (2003). Shallow parsing with conditional random fields. *HLT-NAACL*.

Spitkovsky, V., Alshawi, H., & Jurafsky, D. (2010). From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. *NAACL-HLT*.

Suzuki, J., & Isozaki, H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. *ACL-08: HLT* (pp. 665–673).

Suzuki, J., Isozaki, H., Carreras, X., & Collins, M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. *EMNLP*.

Turian, J., Ratinov, L., Bengio, Y., & Roth, D. (2009). A preliminary evaluation of word representations for named-entity recognition. *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*.

Ushioda, A. (1996). Hierarchical clustering of words. *COLING* (pp. 1159–1162).

Väyrynen, J., & Honkela, T. (2005). Comparison of independent component analysis and singular value decomposition in word context analysis. *AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*.

Väyrynen, J. J., & Honkela, T. (2004). Word category maps based on emergent features created by ICA. *Proceedings of the STeP'2004 Cognition + Cybernetics Symposium* (pp. 173–185). Finnish Artificial Intelligence Society.

Väyrynen, J. J., Honkela, T., & Lindqvist, L. (2007). Towards explicit semantic features using independent component analysis. *Proceedings of the Workshop Semantic Content Acquisition and Representation (SCAR)*. Stockholm, Sweden: Swedish Institute of Computer Science.

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. *LREC*.

Zhang, T., & Johnson, D. (2003). A robust risk minimization based named entity recognition system. *CoNLL*.

Zhao, H., Chen, W., Kit, C., & Zhou, G. (2009). Multilingual dependency learning: a huge feature engineering method to semantic dependency parsing. *CoNLL* (pp. 55–60).

Identifying Text Polarity Using Random Walks

Ahmed Hassan
University of Michigan Ann Arbor
Ann Arbor, Michigan, USA
hassanam@umich.edu

Dragomir Radev
University of Michigan Ann Arbor
Ann Arbor, Michigan, USA
radev@umich.edu

Abstract

Automatically identifying the polarity of words is a very important task in Natural Language Processing. It has applications in text classification, text filtering, analysis of product review, analysis of responses to surveys, and mining online discussions. We propose a method for identifying the polarity of words. We apply a Markov random walk model to a large word relatedness graph, producing a polarity estimate for any given word. A key advantage of the model is its ability to accurately and quickly assign a polarity sign and magnitude to any word. The method could be used both in a semi-supervised setting where a training set of labeled words is used, and in an unsupervised setting where a handful of seeds is used to define the two polarity classes. The method is experimentally tested using a manually labeled set of positive and negative words. It outperforms the state of the art methods in the semi-supervised setting. The results in the unsupervised setting is comparable to the best reported values. However, the proposed method is faster and does not need a large corpus.

1 Introduction

Identifying emotions and attitudes from unstructured text is a very important task in Natural Language Processing. This problem has a variety of possible applications. For example, there has been a great body of work for mining product reputation on the Web (Morinaga et al., 2002; Turney, 2002). Knowing the reputation of a product is very important for marketing and customer relation management (Morinaga et al., 2002). Manually handling reviews to identify reputation is a very costly, and time consuming process given the overwhelming amount of reviews on the Web. A list of words with positive/negative polarity is a very valuable resource for such an application.

Another interesting application is mining online discussions. A threaded discussion is an electronic discussion in which software tools are used to help individuals post messages and respond to other messages. Threaded discussions include e-mails, e-mail lists, bulletin boards, newsgroups, or Internet forums. Threaded discussions act as a very important tool for communication and collaboration in the Web. An enormous number of discussion groups exists on the Web. Millions of users post content to these groups covering pretty much every possible topic. Tracking participant attitude towards different topics and towards other participants is a very interesting task. For example,Tong (2001) presented the concept of sentiment timelines. His system classifies discussion posts about movies as either positive or negative. This is used to produce a plot of the number of positive and negative sentiment messages over time. All those applications could benefit much from an automatic way of identifying semantic orientation of words.

In this paper, we study the problem of automatically identifying semantic orientation of any word by analyzing its relations to other words. Automatically classifying words as either positive or negative enables us to automatically identify the polarity of larger pieces of text. This could be a very useful building block for mining surveys, product reviews and online discussions. We apply a Markov random walk model to a large semantic word graph, producing a polarity estimate for any given word. Previous work on identifying the semantic orientation of words has addressed the problem as both a semi-supervised (Takamura et al., 2005) and an unsupervised (Turney and Littman, 2003) learning problem. In the semi-supervised setting, a training set of labeled words

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 395–403,
Uppsala, Sweden, 11-16 July 2010.

is used to train the model. In the unsupervised setting, only a handful of seeds is used to define the two polarity classes. The proposed method could be used both in a semi-supervised and in an unsupervised setting. Empirical experiments on a labeled set of words show that the proposed method outperforms the state of the art methods in the semi-supervised setting. The results in the unsupervised setting are comparable to the best reported values. The proposed method has the advantages that it is faster and it does not need a large training corpus.

The rest of the paper is structured as follows. In Section 2, we discuss related work. Section 3 presents our method for identifying word polarity. Section 4 describes our experimental setup. We conclude in Section 5.

2 Related Work

Hatzivassiloglou and McKeown (1997) proposed a method for identifying word polarity of adjectives. They extract all conjunctions of adjectives from a given corpus and then they classify each conjunctive expression as either the same orientation such as "simple and well-received" or different orientation such as "simplistic but well-received". The result is a graph that they cluster into two subsets of adjectives. They classify the cluster with the higher average frequency as positive. They created and labeled their own dataset for experiments. Their approach will probably works only with adjectives because there is nothing wrong with conjunctions of nouns or verbs with opposite polarities (e.g., "war and peace", "rise and fall", ..etc).

Turney and Littman (2003) identify word polarity by looking at its statistical association with a set of positive/negative seed words. They use two statistical measures for estimating association: Pointwise Mutual Information (PMI) and Latent Semantic Analysis (LSA). To get co-occurrence statistics, they submit several queries to a search engine. Each query consists of the given word and one of the seed words. They use the search engine near operator to look for instances where the given word is physically close to the seed word in the returned document. They present their method as an unsupervised method where a very small amount of seed words are used to define semantic orientation rather than train the model. One of the limitations of their method is that it requires a large corpus of text to achieve good performance. They use several corpora, the size of the best performing dataset is roughly one hundred billion words (Turney and Littman, 2003).

Takamura et al. (2005) proposed using spin models for extracting semantic orientation of words. They construct a network of words using gloss definitions, thesaurus, and co-occurrence statistics. They regard each word as an electron. Each electron has a spin and each spin has a direction taking one of two values: up or down. Two neighboring spins tend to have the same orientation from an energetic point of view. Their hypothesis is that as neighboring electrons tend to have the same spin direction, neighboring words tend to have similar polarity. They pose the problem as an optimization problem and use the mean field method to find the best solution. The analogy with electrons leads them to assume that each word should be either positive or negative. This assumption is not accurate because most of the words in the language do not have any semantic orientation. They report that their method could get misled by noise in the gloss definition and their computations sometimes get trapped in a local optimum because of its greedy optimization flavor.

Kamps et al. (2004) construct a network based on WordNet synonyms and then use the shortest paths between any given word and the words 'good' and 'bad' to determine word polarity. They report that using shortest paths could be very noisy. For example. 'good' and 'bad' themselves are closely related in WordNet with a 5-long sequence "good, sound, heavy, big, bad". A given word w may be more connected to one set of words (e.g., positive words), yet have a shorter path connecting it to one word in the other set. Restricting seed words to only two words affects their accuracy. Adding more seed words could help but it will make their method extremely costly from the computation point of view. They evaluate their method only using adjectives.

Hu and Liu (2004) use WordNet synonyms and antonyms to predict the polarity of words. For any word, whose polarity is unknown, they search WordNet and a list of seed labeled words to predict its polarity. They check if any of the synonyms of the given word has known polarity. If so, they label it with the label of its synonym. Otherwise, they check if any of the antonyms of the given word has known polarity. If so, they label it

with the opposite label of the antonym. They continue in a bootstrapping manner till they label all possible word. This method is quite similar to the shortest-path method proposed in (Kamps et al., 2004).

There are some other methods that try to build lexicons of polarized words. Esuli and Sebastiani (2005; 2006) use a textual representation of words by collating all the glosses of the word as found in some dictionary. Then, a binary text classifier is trained using the textual representation and applied to new words. Kim and Hovy (2004) start with two lists of positive and negative seed words. WordNet is used to expand these lists. Synonyms of positive words and antonyms of negative words are considered positive, while synonyms of negative words and antonyms of positive words are considered negative. A similar method is presented in (Andreevskaia and Bergler, 2006) where WordNet synonyms, antonyms, and glosses are used to iteratively expand a list of seeds. The sentiment classes are treated as fuzzy categories where some words are very central to one category, while others may be interpreted differently. Kanayama and Nasukawa (2006) use syntactic features and context coherency, the tendency for same polarities to appear successively , to acquire polar atoms.

Other related work is concerned with subjectivity analysis. Subjectivity analysis is the task of identifying text that present opinions as opposed to objective text that present factual information (Wiebe, 2000). Text could be either words, phrases, sentences, or any other chunks. There are two main categories of work on subjectivity analysis. In the first category, subjective words and phrases are identified without considering their context (Wiebe, 2000; Hatzivassiloglou and Wiebe, 2000; Banea et al., 2008). In the second category, the context of subjective text is used (Riloff and Wiebe, 2003; Yu and Hatzivassiloglou, 2003; Nasukawa and Yi, 2003; Popescu and Etzioni, 2005) Wiebe et al. (2001) lists a lot of applications of subjectivity analysis such as classifying emails and mining reviews. Subjectivity analysis is related to the proposed method because identifying the polarity of text is the natural next step that should follow identifying subjective text.

3 Word Polarity

We use a Markov random walk model to identify polarity of words. Assume that we have a network of words, some of which are labeled as either positive or negative. In this network, two words are connecting if they are related. Different sources of information could be used to decide whether two words are related or not. For example, the synonyms of any word are semantically related to it. The intuition behind that connecting semantically related words is that those words tend to have similar polarity. Now imagine a random surfer walking along the network starting from an unlabeled word w. The random walk continues until the surfer hits a labeled word. If the word w is positive then the probability that the random walk hits a positive word is higher and if w is negative then the probability that the random walk hits a negative word is higher. Similarly, if the word w is positive then the average time it takes a random walk starting at w to hit a positive node is less than the average time it takes a random walk starting at w to hit a negative node.

In the rest of this section, we will describe how we can construct a word relatedness graph in Section 3.1. The random walk model is described in Section 3.2. Hitting time is defined in Section‘3.3. Finally, an algorithm for computing a sign and magnitude for the polarity of any given word is described in Section 3.4.

3.1 Network Construction

We construct a network where two nodes are linked if they are semantically related. Several sources of information could be used as indicators of the relatedness of words. One such important source is WordNet (Miller, 1995). WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept (Miller, 1995). Synsets are interlinked by means of conceptual-semantic and lexical relations.

The simplest approach is to connect words that occur in the same WordNet synset. We can collect all words in WordNet, and add links between any two words that occurr in the same synset. The resulting graph is a graph $G(W, E)$ where W is a set of word / part-of-speech pairs for all the words in WordNet. E is the set of edges connecting each pair of synonymous words. Nodes represent word/pos pairs rather than words because the part of speech tags are helpful in disambiguating the different senses for a given word. For example,

the word "fine" has two different meanings when used as an adjective and as a noun.

Several other methods could be used to link words. For example, we can use other WordNet relations: hypernyms, similar to,...etc. Another source of links between words is co-occurrence statistics from corpus. Following the method presented in (Hatzivassiloglou and McKeown, 1997), we can connect words if they appear in a conjunctive form in the corpus. This method is only applicable to adjectives. If two adjectives are connected by "and" in conjunctive form, it is highly likely that they have the same semantic orientation. In all our experiments, we restricted the network to only WordNet relations. We study the effect of using co-occurrence statistics to connect words later at the end of our experiments. If more than one relation exists between any two words, the strength of the corresponding edge is adjusted accordingly.

3.2 Random Walk Model

Imagine a random surfer walking along the word relatedness graph G. Starting from a word with unknown polarity i , it moves to a node j with probability P_{ij} after the first step. The walk continues until the surfer hits a word with a known polarity. Seed words with known polarity act as an absorbing boundary for the random walk. If we repeat the number of random walks N times, the percentage of time at which the walk ends at a positive/negative word could be used as an indicator of its positive/negative polarity. The average time a random walk starting at w takes to hit the set of positive/negative nodes is also an indicator of its polarity. This view is closely related to the partially labeled classification with random walks approach in (Szummer and Jaakkola, 2002) and the semi-supervised learning using harmonic functions approach in (Zhu et al., 2003).

Let W be the set of words in our lexicon. We construct a graph whose nodes V are all words in W The edges E correspond to relatedness between words We define transition probabilities $P_{t+1|t}(j|i)$ from i to j by normalizing the weights of the edges out of node i, so:

$$P_{t+1|t}(j|i) = Wij / \sum_k W_{ik} \tag{1}$$

where k represents all nodes in the neighborhood of i. $P_{t2|t1}(j|i)$ denotes the transition probability from node i at step t_1 to node j at time step t_2. We note that the weights W_{ij} are symmetric and the transition probabilities $P_{t+1|t}(j|i)$ are not necessarily symmetric because of the node out degree normalization.

3.3 First-Passage Time

The mean first-passage (hitting) time $h(i|k)$ is defined as the average number of steps a random walker, starting in state $i \neq k$, will take to enter state k for the first time (Norris, 1997). Let $G = (V, E)$ be a graph with a set of vertices V, and a set of edges E. Consider a subset of vertices $S \subset V$, Consider a random walk on G starting at node $i \notin S$. Let N_t denote the position of the random surfer at time t. Let $h(i|S)$ be the the average number of steps a random walker, starting in state $i \notin S$, will take to enter a state $k \in S$ for the first time. Let T^S be the first-passage for any vertex in S.

$$P(T_S = t|N_0 = i) = \sum_{j \in V} p_{ij} \times P(T_S = t-1|N_0 = j) \tag{2}$$

$h(i|S)$ is the expectation of T_S. Hence:

$$\begin{aligned}
h(i|S) &= E(T_S|N_0 = i) \\
&= \sum_{t=1}^{\infty} t \times P(T_S = t|N_0 = i) \\
&= \sum_{t=1}^{\infty} t \sum_{j \in V} p_{ij} P(T_S = t-1|N_0 = j) \\
&= \sum_{j \in V} \sum_{t=1}^{\infty} (t-1) p_{ij} P(T_S = t-1|N_0 = j) \\
&+ \sum_{j \in V} \sum_{t=1}^{\infty} p_{ij} P(T_S = t-1|N_0 = j) \\
&= \sum_{j \in V} p_{ij} \sum_{t=1}^{\infty} t P(T_S = t|N_0 = j) + 1 \\
&= \sum_{j \in V} p_{ij} \times h(j|S) + 1
\end{aligned} \tag{3}$$

Hence the first-passage (hitting) time can be formally defined as:

$$h(i|S) = \begin{cases} 0 & i \in S \\ \sum_{j \in V} p_{ij} \times h(j|S) + 1 & \text{otherwise} \end{cases} \tag{4}$$

3.4 Word Polarity Calculation

Based on the description of the random walk model and the first-passage (hitting) time above,

we now propose our word polarity identification algorithm. We begin by constructing a word relatedness graph and defining a random walk on that graph as described above. Let S^+ and S^- be two sets of vertices representing seed words that are already labeled as either positive or negative respectively. For any given word w, we compute the hitting time $h(w|S^+)$, and $h(w|S^-)$ for the two sets iteratively as described earlier. if $h(w|S^+)$ is greater than $h(w|S^-)$, the word is classified as negative, otherwise it is classified as positive. The ratio between the two hitting times could be used as an indication of how positive/negative the given word is. This is useful in case we need to provide a confidence measure for the prediction. This could be used to allow the model to abstain from classifying words with when the confidence level is low.

Computing hitting time as described earlier may be time consuming especially if the graph is large. To overcome this problem, we propose a Monte Carlo based algorithm for estimating it. The algorithm is shown in Algorithm 1.

Algorithm 1 Word Polarity using Random Walks

Require: A word relatedness graph G

1: Given a word w in V
2: Define a random walk on the graph. the transition probability between any two nodes i, and j is defined as: $P_{t+1|t}(j|i) = Wij / \sum_k W_{ik}$
3: Start k independent random walks from w with a maximum number of steps m
4: Stop when a positive word is reached
5: Let $h^*(w|S^+)$ be the estimated value for $h(w|S^+)$
6: Repeat for negative words computing $h^*(w|S^-)$
7: **if** $h^*(w|S^+) \leq h^*(w|S^-)$ **then**
8: Classify w as positive
9: **else**
10: Classify w as negative
11: **end if**

4 Experiments

We performed experiments on the General Inquirer lexicon (Stone et al., 1966). We used it as a gold standard data set for positive/negative words. The dataset contains 4206 words, 1915 of which are positive and 2291 are negative. Some of the ambiguous words were removed like (Turney, 2002; Takamura et al., 2005).

We use WordNet (Miller, 1995) as a source of synonyms and hypernyms for the word relatedness graph. We used 10-fold cross validation for all tests. We evaluate our results in terms of accuracy. Statistical significance was tested using a 2-tailed paired t-test. All reported results are statistically significant at the 0.05 level. We perform experiments varying the parameters and the network. We also look at the performance of the proposed method for different parts of speech, and for different confidence levels We compare our method to the Semantic Orientation from PMI (SO-PMI) method described in (Turney, 2002), the Spin model (Spin) described in (Takamura et al., 2005), the shortest path (short-path) described in (Kamps et al., 2004), and the bootstrapping (bootstrap) method described in (Hu and Liu, 2004).

4.1 Comparisons with other methods

This method could be used in a semi-supervised setting where a set of labeled words are used and the system learns from these labeled nodes and from other unlabeled nodes. Under this setting, we compare our method to the spin model described in (Takamura et al., 2005). Table 2 compares the performance using 10-fold cross validation. The table shows that the proposed method outperforms the spin model. The spin model approach uses word glosses, WordNet synonym, hypernym, and antonym relations, in addition to co-occurrence statistics extracted from corpus. The proposed method achieves better performance by only using WordNet synonym, hypernym and similar to relations. Adding co-occurrence statistics slightly improved performance, while using glosses did not help at all.

We also compare our method to the SO-PMI method presented in (Turney, 2002). They describe this setting as unsupervised (Turney, 2002) because they only use 14 seeds as paradigm words that define the semantic orientation rather than train the model. After (Turney, 2002), we use our method to predict semantic orientation of words in the General Inquirer lexicon (Stone et al., 1966) using only 14 seed words. The network we used contains only WordNet relations. No glosses or co-occurrence statistics are used. The results comparing the SO-PMI method with different dataset sizes, the spin model, and the proposed method using only 14 seeds is shown in Table 2. We no-

Table 1: Accuracy for adjectives only for the spin model, the bootstrap method, and the random walk model.

spin-model	bootstrap	short-path	rand-walks
83.6	72.8	68.8	88.8

tice that the random walk method outperforms SO-PMI when SO-PMI uses datasets of sizes 1×10^7 and 2×10^9 words. The performance of SO-PMI and the random walk methods are comparable when SO-PMI uses a very large dataset (1×10^{11} words). The performance of the spin model approach is also comparable to the other 2 methods. The advantages of the random walk method over SO-PMI is that it is faster and it does not need a very large corpus like the one used by SO-PMI. Another advantage is that the random walk method can be used along with the labeled data from the General Inquirer lexicon (Stone et al., 1966) to get much better performance. This is costly for the SO-PMI method because that will require the submission of almost 4000 queries to a commercial search engine.

We also compare our method to the bootstrapping method described in (Hu and Liu, 2004), and the shortest path method described in (Kamps et al., 2004). We build a network using only WordNet synonyms and hypernyms. We restrict the test set to the set of adjectives in the General Inquirer lexicon (Stone et al., 1966) because this method is mainly interested in classifying adjectives. The performance of the spin model method, the bootstrapping method, the shortest path method, and the random walk method for only adjectives is shown in Table 1. We notice from the table that the random walk method outperforms both the spin model, the bootstrapping method, and the shortest path method for adjectives. The reported accuracy for the shortest path method only considers the words it could assign a non-zero orientation value. If we consider all words, the accuracy will drop to around 61%.

4.1.1 Varying Parameters

As we mentioned in Section 3.4, we use a parameter m to put an upper bound on the length of random walks. In this section, we explore the impact of this parameter on our method's performance.

Table 2: Accuracy for SO-PMI with different dataset sizes, the spin model, and the random walks model for 10-fold cross validation and 14 seeds.

-	CV	14 seeds
SO-PMI (1×10^7)	-	61.3
SO-PMI (2×10^9)	-	76.1
SO-PMI (1×10^{11})	-	**82.8**
Spin Model	91.5	81.9
Random Walks	**93.1**	82.1

Figure 1 shows the accuracy of the random walk method as a function of the maximum number of steps m. m varies from 5 to 50. We use a network built from WordNet synonyms and hypernyms only. The number of samples k was set to 1000. We perform 10-fold cross validation using the General Inquirer lexicon. We notice that the maximum number of steps m has very little impact on performance until it rises above 30. When it does, the performance drops by no more than 1%, and then it does not change anymore as m increases. An interesting observation is that the proposed method performs quite well with a very small number of steps (around 10). We looked at the dataset to understand why increasing the number of steps beyond 30 negatively affects performance. We found out that when the number of steps is very large, compared to the diameter of the graph, the random walk that starts at ambiguous words, that are hard to classify, have the chance of moving till it hits a node in the opposite class. That does not happen when the limit on the number of steps is smaller because those walks are then terminated without hitting any labeled nodes and hence ignored.

Next, we study the effect of the random of samples k on our method's performance. As explained in Section 3.4, k is the number of samples used by the Monte Carlo algorithm to find an estimate for the hitting time. Figure 2 shows the accuracy of the random walks method as a function of the number of samples k. We use the same settings as in the previous experiment. the only difference is that we fix m at 15 and vary k from 10 to 20000 (note the logarithmic scale). We notice that the performance is badly affected, when the value of k is very small (less than 100). We also notice that

after 1000, varying k has very little, if any, effect on performance. This shows that the Monte Carlo algorithm for computing the random walks hitting time performs quite well with values of the number of samples as small as 1000.

The preceding experiments suggest that the parameter have very little impact on performance. This suggests that the approach is fairly robust (i.e., it is quite insensitive to different parameter settings).

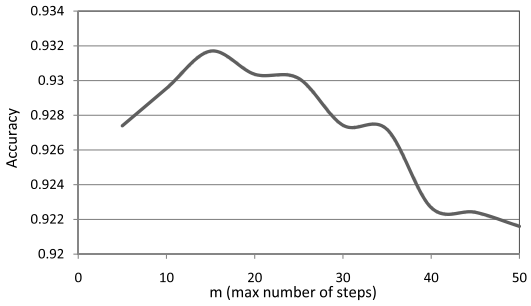


Figure 1: The effect of varying the maximum number of steps (m) on accuracy.

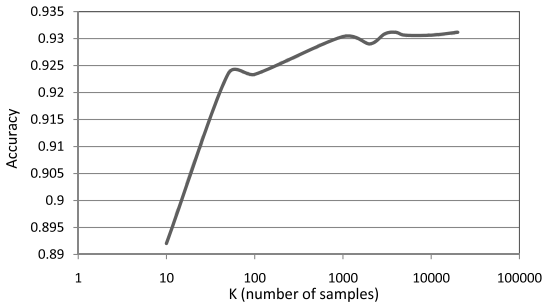


Figure 2: The effect of varying the number of samples (k) on accuracy.

4.1.2 Other Experiments

We now measure the performance of the proposed method when the system is allowed to abstain from classifying the words for which it have low confidence. We regard the ratio between the hitting time to positive words and hitting time to negative words as a confidence measure and evaluate the top words with the highest confidence level at different values of threshold. Figure 4 shows the accuracy for 10-fold cross validation and for using only 14 seeds at different thresholds. We notice that the accuracy improves by abstaining from classifying the difficult words. The figure shows that the top 60% words are classified with an accuracy greater than 99% for 10-fold cross validation and 92% with 14 seed words. This may be compared to the work descibed in (Takamura et al., 2005) where they achieve the 92% level when they only consider the top 1000 words (28%).

Figure 3 shows a learning curve displaying how the performance of the proposed method is affected with varying the labeled set size (i.e., the number of seeds). We notice that the accuracy exceeds 90% when the training set size rises above 20%. The accuracy steadily increases as the labeled data increases.

We also looked at the classification accuracy for different parts of speech in Figure 5. we notice that, in the case of 10-fold cross validation, the performance is consistent across parts of speech. However, when we only use 14 seeds all of which are adjectives, similar to (Turney and Littman, 2003), we notice that the performance on adjectives is much better than other parts of speech. When we use 14 seeds but replace some of the adjectives with verbs and nouns like (love, harm, friend, enemy), the performance for nouns and verbs improves considerably at the cost of losing a little bit of the performance on adjectives. We had a closer look at the results to find out what are the reasons behind incorrect predictions. We found two main reasons. First, some words are ambiguous and has more than one sense, possible with different orientations. Disambiguating the sense of words given their context before trying to predict their polarity should solve this problem. The second reason is that some words have very few connection in thesaurus. A possible solution to this might be identifying those words and adding more links to them from glosses of co-occurrence statistics in corpus.

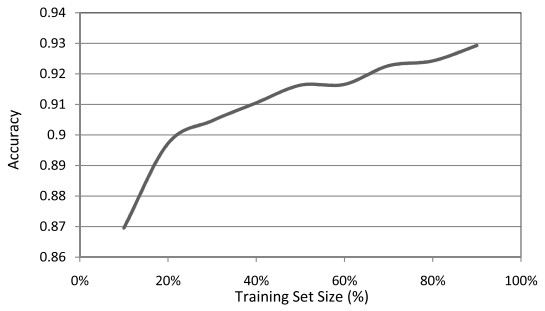


Figure 3: The effect of varying the number of seeds on accuracy.

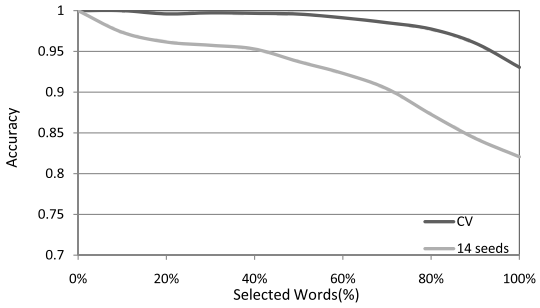


Figure 4: Accuracy for words with high confidence measure.

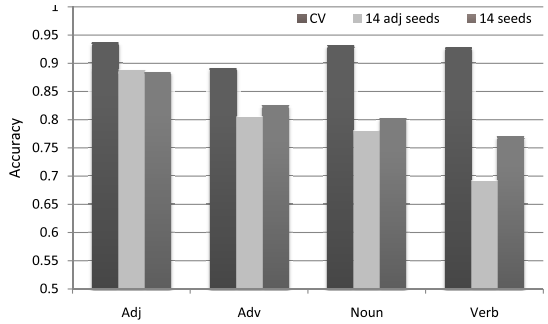


Figure 5: Accuracy for different parts of speech.

5 Conclusions

Predicting the semantic orientation of words is a very interesting task in Natural Language Processing and it has a wide variety of applications. We proposed a method for automatically predicting the semantic orientation of words using random walks and hitting time. The proposed method is based on the observation that a random walk starting at a given word is more likely to hit another word with the same semantic orientation before hitting a word with a different semantic orientation. The proposed method can be used in a semi-supervised setting where a training set of labeled words is used, and in an unsupervised setting where only a handful of seeds is used to define the two polarity classes. We predict semantic orientation with high accuracy. The proposed method is fast, simple to implement, and does not need any corpus.

Acknowledgments

This research was funded by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Alina Andreevskaia and Sabine Bergler. 2006. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th Conference on Information and Knowledge Management (CIKM 2005)*, pages 617–624.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC 2006)*, pages 417–422.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181.

Vasileios Hatzivassiloglou and Janyce Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *COLING*, pages 299–305.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientations of adjectives. In *National Institute for*, pages 1115–1118.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 355–363.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics (COLING 2004)*, pages 1367–1373.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. 2002. Mining product reputations on the web. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 341–349.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: capturing favorability using natural language processing. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77.

J. Norris. 1997. Markov chains. Cambridge University Press.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112.

Philip Stone, Dexter Dunphy, Marchall Smith, and Daniel Ogilvie. 1966. The general inquirer: A computer approach to content analysis. *The MIT Press*.

Martin Szummer and Tommi Jaakkola. 2002. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 133–140.

Richard M. Tong. 2001. An operational system for detecting and tracking opinions in on-line discussion. Workshop note, SIGIR 2001 Workshop on Operational Text Classification.

Peter Turney and Michael Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21:315–346.

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

Janyce Wiebe, Rebecca Bruce, Matthew Bell, Melanie Martin, and Theresa Wilson. 2001. A corpus study of evaluative and speculative language. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1–10.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *In ICML*, pages 912–919.

Sentiment Learning on Product Reviews via Sentiment Ontology Tree

Wei Wei
Department of Computer and Information Science
Norwegian University of Science and Technology
wwei@idi.ntnu.no

Jon Atle Gulla
Department of Computer and Information Science
Norwegian University of Science and Technology
jag@idi.ntnu.no

Abstract

Existing works on sentiment analysis on product reviews suffer from the following limitations: (1) The knowledge of hierarchical relationships of products attributes is not fully utilized. (2) Reviews or sentences mentioning several attributes associated with complicated sentiments are not dealt with very well. In this paper, we propose a novel HL-SOT approach to labeling a product's attributes and their associated sentiments in product reviews by a Hierarchical Learning (HL) process with a defined Sentiment Ontology Tree (SOT). The empirical analysis against a human-labeled data set demonstrates promising and reasonable performance of the proposed HL-SOT approach. While this paper is mainly on sentiment analysis on reviews of one product, our proposed HL-SOT approach is easily generalized to labeling a mix of reviews of more than one products.

1 Introduction

As the internet reaches almost every corner of this world, more and more people write reviews and share opinions on the World Wide Web. The user-generated opinion-rich reviews will not only help other users make better judgements but they are also useful resources for manufacturers of products to keep track and manage customer opinions. However, as the number of product reviews grows, it becomes difficult for a user to manually learn the panorama of an interesting topic from existing online information. Faced with this problem, research works, e.g., (Hu and Liu, 2004; Liu et al., 2005; Lu et al., 2009), of sentiment analysis on product reviews were proposed and have become a popular research topic at the crossroads of information retrieval and computational linguistics.

Carrying out sentiment analysis on product reviews is not a trivial task. Although there have already been a lot of publications investigating on similar issues, among which the representatives are (Turney, 2002; Dave et al., 2003; Hu and Liu, 2004; Liu et al., 2005; Popescu and Etzioni, 2005; Zhuang et al., 2006; Lu and Zhai, 2008; Titov and McDonald, 2008; Zhou and Chaovalit, 2008; Lu et al., 2009), there is still room for improvement on tackling this problem. When we look into the details of each example of product reviews, we find that there are some intrinsic properties that existing previous works have not addressed in much detail.

First of all, product reviews constitute domain-specific knowledge. The product's attributes mentioned in reviews might have some relationships between each other. For example, for a digital camera, comments on image quality are usually mentioned. However, a sentence like "40D handles noise very well up to ISO 800", also refers to image quality of the camera 40D. Here we say "noise" is a sub-attribute factor of "image quality". We argue that the hierarchical relationship between a product's attributes can be useful knowledge if it can be formulated and utilized in product reviews analysis. Secondly, Vocabularies used in product reviews tend to be highly overlapping. Especially, for same attribute, usually same words or synonyms are involved to refer to them and to describe sentiment on them. We believe that labeling existing product reviews with attributes and corresponding sentiment forms an effective training resource to perform sentiment analysis. Thirdly, sentiments expressed in a review or even in a sentence might be opposite on different attributes and not every attributes mentioned are with sentiments. For example, it is common to find a fragment of a review as follows:
Example 1: *"...I am very impressed with this camera except for its a bit heavy weight especially with*

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 404–413,
Uppsala, Sweden, 11-16 July 2010.

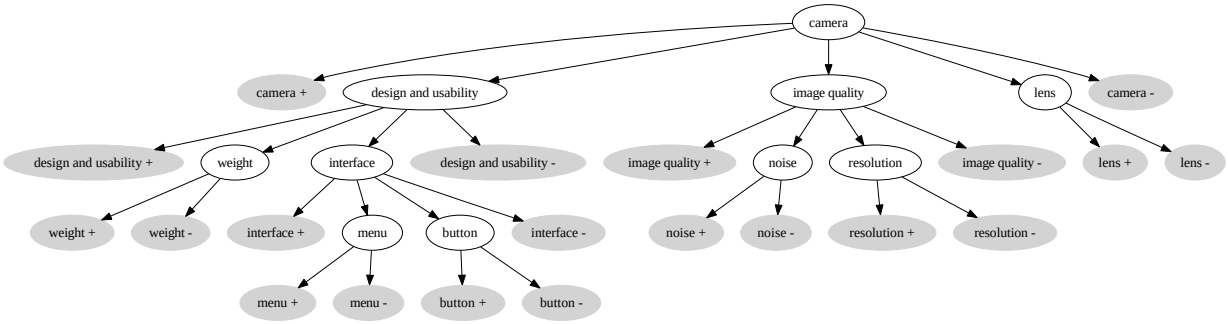


Figure 1: an example of part of a SOT for digital camera

extra lenses attached. It has many buttons and two main dials. The first dial is thumb dial, located near shutter button. The second one is the big round dial located at the back of the camera..."
In this example, the first sentence gives positive comment on the camera as well as a complaint on its heavy weight. Even if the words "lenses" appears in the review, it is not fair to say the customer expresses any sentiment on lens. The second sentence and the rest introduce the camera's buttons and dials. It's also not feasible to try to get any sentiment from these contents. We argue that when performing sentiment analysis on reviews, such as in the Example 1, more attention is needed to distinguish between attributes that are mentioned with and without sentiment.

In this paper, we study the problem of sentiment analysis on product reviews through a novel method, called the HL-SOT approach, namely Hierarchical Learning (HL) with Sentiment Ontology Tree (SOT). By sentiment analysis on product reviews we aim to fulfill two tasks, i.e., labeling a target text[1] with: 1) the product's attributes (attributes identification task), and 2) their corresponding sentiments mentioned therein (sentiment annotation task). The result of this kind of labeling process is quite useful because it makes it possible for a user to search reviews on particular attributes of a product. For example, when considering to buy a digital camera, a prospective user who cares more about image quality probably wants to find comments on the camera's image quality in other users' reviews. SOT is a tree-like ontology structure that formulates the relationships between a product's attributes. For example, Fig. 1 is a SOT for a digital camera[2]. The root node of the SOT is a camera itself. Each of the non-leaf nodes (white nodes) of the SOT represents an attribute of a camera[3]. All leaf nodes (gray nodes) of the SOT represent sentiment (positive/negative) nodes respectively associated with their parent nodes. A formal definition on SOT is presented in Section 3.1. With the proposed concept of SOT, we manage to formulate the two tasks of the sentiment analysis to be a hierarchical classification problem. We further propose a specific hierarchical learning algorithm, called HL-SOT algorithm, which is developed based on generalizing an online-learning algorithm H-RLS (Cesa-Bianchi et al., 2006). The HL-SOT algorithm has the same property as the H-RLS algorithm that allows multiple-path labeling (input target text can be labeled with nodes belonging to more than one path in the SOT) and partial-path labeling (the input target text can be labeled with nodes belonging to a path that does not end on a leaf). This property makes the approach well suited for the situation where complicated sentiments on different attributes are expressed in one target text. Unlike the H-RLS algorithm , the HL-SOT algorithm enables each classifier to separately learn its own specific threshold. The proposed HL-SOT approach is empirically analyzed against a human-labeled data set. The experimental results demonstrate promising and reasonable performance of our approach.

This paper makes the following contributions:

- To the best of our knowledge, with the proposed concept of SOT, the proposed HL-SOT approach is the first work to formulate the tasks of sentiment analysis to be a hierarchical classification problem.

- A specific hierarchical learning algorithm is

[1]Each product review to be analyzed is called target text in the following of this paper.

[2]Due to the space limitation, not all attributes of a digital camera are enumerated in this SOT; m+/m- means positive/negative sentiment associated with an attribute m.

[3]A product itself can be treated as an overall attribute of the product.

further proposed to achieve tasks of sentiment analysis in one hierarchical classification process.

- The proposed HL-SOT approach can be generalized to make it possible to perform sentiment analysis on target texts that are a mix of reviews of different products, whereas existing works mainly focus on analyzing reviews of only one type of product.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of related work on sentiment analysis. Section 3 presents our work on sentiment analysis with HL-SOT approach. The empirical analysis and the results are presented in Section 4, followed by the conclusions, discussions, and future work in Section 5.

2 Related Work

The task of sentiment analysis on product reviews was originally performed to extract overall sentiment from the target texts. However, in (Turney, 2002), as the difficulty shown in the experiments, the whole sentiment of a document is not necessarily the sum of its parts. Then there came up with research works shifting focus from overall document sentiment to sentiment analysis based on product attributes (Hu and Liu, 2004; Popescu and Etzioni, 2005; Ding and Liu, 2007; Liu et al., 2005).

Document overall sentiment analysis is to summarize the overall sentiment in the document. Research works related to document overall sentiment analysis mainly rely on two finer levels sentiment annotation: *word-level sentiment annotation* and *phrase-level sentiment annotation*. The *word-level sentiment annotation* is to utilize the polarity annotation of words in each sentence and summarize the overall sentiment of each sentiment-bearing word to infer the overall sentiment within the text (Hatzivassiloglou and Wiebe, 2000; Andreevskaia and Bergler, 2006; Esuli and Sebastiani, 2005; Esuli and Sebastiani, 2006; Hatzivassiloglou and McKeown, 1997; Kamps et al., 2004; Devitt and Ahmad, 2007; Yu and Hatzivassiloglou, 2003). The *phrase-level sentiment annotation* focuses sentiment annotation on phrases not words with concerning that atomic units of expression is not individual words but rather appraisal groups (Whitelaw et al., 2005). In (Wilson et al., 2005), the concepts of *prior polarity* and *contextual polarity* were proposed. This paper presented a system that is able to automatically identify the *contextual polarity* for a large subset of sentiment expressions. In (Turney, 2002), an unsupervised learning algorithm was proposed to classify reviews as recommended or not recommended by averaging sentiment annotation of phrases in reviews that contain adjectives or adverbs. However, the performances of these works are not good enough for sentiment analysis on product reviews, where sentiment on each attribute of a product could be so complicated that it is unable to be expressed by overall document sentiment.

Attributes-based sentiment analysis is to analyze sentiment based on each attribute of a product. In (Hu and Liu, 2004), mining product features was proposed together with sentiment polarity annotation for each opinion sentence. In that work, sentiment analysis was performed on product attributes level. In (Liu et al., 2005), a system with framework for analyzing and comparing consumer opinions of competing products was proposed. The system made users be able to clearly see the strengths and weaknesses of each product in the minds of consumers in terms of various product features. In (Popescu and Etzioni, 2005), Popescu and Etzioni not only analyzed polarity of opinions regarding product features but also ranked opinions based on their strength. In (Liu et al., 2007), Liu et al. proposed Sentiment-PLSA that analyzed blog entries and viewed them as a document generated by a number of hidden sentiment factors. These sentiment factors may also be factors based on product attributes. In (Lu and Zhai, 2008), Lu et al. proposed a semi-supervised topic models to solve the problem of opinion integration based on the topic of a product's attributes. The work in (Titov and McDonald, 2008) presented a multi-grain topic model for extracting the ratable attributes from product reviews. In (Lu et al., 2009), the problem of rated attributes summary was studied with a goal of generating ratings for major aspects so that a user could gain different perspectives towards a target entity. All these research works concentrated on attribute-based sentiment analysis. However, the main difference with our work is that they did not sufficiently utilize the hierarchical relationships among a product attributes. Although a method of ontology-supported polarity mining, which also involved

ontology to tackle the sentiment analysis problem, was proposed in (Zhou and Chaovalit, 2008), that work studied polarity mining by machine learning techniques that still suffered from a problem of ignoring dependencies among attributes within an ontology's hierarchy. In the contrast, our work solves the sentiment analysis problem as a hierarchical classification problem that fully utilizes the hierarchy of the SOT during training and classification process.

3 The HL-SOT Approach

In this section, we first propose a formal definition on SOT. Then we formulate the HL-SOT approach. In this novel approach, tasks of sentiment analysis are to be achieved in a hierarchical classification process.

3.1 Sentiment Ontology Tree

As we discussed in Section 1, the hierarchial relationships among a product's attributes might help improve the performance of attribute-based sentiment analysis. We propose to use a tree-like ontology structure SOT, i.e., Sentiment Ontology Tree, to formulate relationships among a product's attributes. Here,we give a formal definition on what a SOT is.

Definition 1 *[SOT] SOT is an abbreviation for Sentiment Ontology Tree that is a tree-like ontology structure* $T(v, v^+, v^-, \mathbb{T})$. *$v$ is the root node of T which represents an attribute of a given product.* v^+ *is a positive sentiment leaf node associated with the attribute* v. v^- *is a negative sentiment leaf node associated with the attribute* v. $\mathbb{T}$ *is a set of subtrees. Each element of* $\mathbb{T}$ *is also a SOT* $T'(v', v'^+, v'^-, \mathbb{T}')$ *which represents a sub-attribute of its parent attribute node.*

By the Definition 1, we define a root of a SOT to represent an attribute of a product. The SOT's two leaf child nodes are sentiment (positive/negative) nodes associated with the root attribute. The SOT recursively contains a set of sub-SOTs where each root of a sub-SOT is a non-leaf child node of the root of the SOT and represent a sub-attribute belonging to its parent attribute. This definition successfully describes the hierarchical relationships among all the attributes of a product. For example, in Fig. 1 the root node of the SOT for a digital camera is its general overview attribute. Comments on a digital camera's general overview attribute appearing in a review might be like "this camera is great". The "camera" SOT has two sentiment leaf child nodes as well as three non-leaf child nodes which are respectively root nodes of sub-SOTs for sub-attributes "design and usability", "image quality", and "lens". These sub-attributes SOTs recursively repeat until each node in the SOT does not have any more non-leaf child node, which means the corresponding attributes do not have any sub-attributes, e.g., the attribute node "button" in Fig. 1.

3.2 Sentiment Analysis with SOT

In this subsection, we present the HL-SOT approach. With the defined SOT, the problem of sentiment analysis is able to be formulated to be a hierarchial classification problem. Then a specific hierarchical learning algorithm is further proposed to solve the formulated problem.

3.2.1 Problem Formulation

In the proposed HL-SOT approach, each target text is to be indexed by a unit-norm vector $x \in \mathcal{X}, \mathcal{X} = \mathbb{R}^d$. Let $\mathscr{Y} = \{1, ..., N\}$ denote the finite set of nodes in SOT. Let $y = \{y_1, ..., y_N\} \in \{0, 1\}^N$ be a label vector to a target text x, where $\forall i \in \mathscr{Y}$:

$$y_i = \begin{cases} 1, & \text{if } x \text{ is labeled by the classifier of node } i, \\ 0, & \text{if } x \text{ is not labeled by the classifier of node } i. \end{cases}$$

A label vector $y \in \{0, 1\}^N$ is said to respect SOT if and only if y satisfies $\forall i \in \mathscr{Y}, \forall j \in \mathcal{A}(i)$: if $y_i = 1$ then $y_j = 1$, where $\mathcal{A}(i)$ represents a set ancestor nodes of i, i.e.,$\mathcal{A}(i) = \{x | ancestor(i, x)\}$. Let $\mathcal{Y}$ denote a set of label vectors that respect SOT. Then the tasks of sentiment analysis can be formulated to be the goal of a hierarchical classification that is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, that is able to label each target text $x \in \mathcal{X}$ with classifier of each node and generating with x a label vector $y \in \mathcal{Y}$ that respects SOT. The requirement of a generated label vector $y \in \mathcal{Y}$ ensures that a target text is to be labeled with a node only if its parent attribute node is labeled with the target text. For example, in Fig. 1 a review is to be labeled with "image quality +" requires that the review should be successively labeled as related to "camera" and "image quality". This is reasonable and consistent with intuition, because if a review cannot be identified to be related to a camera, it is not safe to infer that the review is commenting a camera's image quality with positive sentiment.

3.2.2 HL-SOT Algorithm

The algorithm H-RLS studied in (Cesa-Bianchi et al., 2006) solved a similar hierarchical classification problem as we formulated above. However, the H-RLS algorithm was designed as an online-learning algorithm which is not suitable to be applied directly in our problem setting. Moreover, the algorithm H-RLS defined the same value as the threshold of each node classifier. We argue that if the threshold values could be learned separately for each classifiers, the performance of classification process would be improved. Therefore we propose a specific hierarchical learning algorithm, named HL-SOT algorithm, that is able to train each node classifier in a batch-learning setting and allows separately learning for the threshold of each node classifier.

Defining the f function Let $w_1, ..., w_N$ be weight vectors that define linear-threshold classifiers of each node in SOT. Let $W = (w_1, ..., w_N)^\top$ be an $N \times d$ matrix called weight matrix. Here we generalize the work in (Cesa-Bianchi et al., 2006) and define the hierarchical classification function f as:

$$\hat{y} = f(x) = g(W \cdot x),$$

where $x \in \mathcal{X}, \hat{y} \in \mathcal{Y}$. Let $z = W \cdot x$. Then the function $\hat{y} = g(z)$ on an N-dimensional vector z defines:
$\forall i = 1, ..., N$:

$$\hat{y}_i = \begin{cases} \mathfrak{B}(z_i \geq \theta_i), & \text{if } i \text{ is a root node in SOT} \\ & \text{or } y_j = 1 \text{ for } j = \mathcal{P}(i), \\ 0, & \text{else} \end{cases}$$

where $\mathcal{P}(i)$ is the parent node of i in SOT and $\mathfrak{B}(S)$ is a boolean function which is 1 if and only if the statement S is true. Then the hierarchical classification function f is parameterized by the weight matrix $W = (w_1, ..., w_N)^\top$ and threshold vector $\theta = (\theta_1, ..., \theta_N)^\top$. The hierarchical learning algorithm HL-SOT is proposed for learning the parameters of W and θ.

Parameters Learning for f function Let D denote the training data set: $D = \{(r, l) | r \in \mathcal{X}, l \in \mathcal{Y}\}$. In the HL-SOT learning process, the weight matrix W is firstly initialized to be a 0 matrix, where each row vector w_i is a 0 vector. The threshold vector is initialized to be a 0 vector. Each instance in the training set D goes into the training process. When a new instance r_t is observed, each row vector $w_{i,t}$ of the weight matrix W_t is updated by a regularized least squares estimator given by:

$$w_{i,t} = (I + S_{i,Q(i,t-1)} S_{i,Q(i,t-1)}^\top + r_t r_t^\top)^{-1} \times S_{i,Q(i,t-1)} (l_{i,i_1}, l_{i,i_2}, ..., l_{i,i_{Q(i,t-1)}})^\top \quad (1)$$

where I is a $d \times d$ identity matrix, $Q(i, t-1)$ denotes the number of times the parent of node i observes a positive label before observing the instance r_t, $S_{i,Q(i,t-1)} = [r_{i_1}, ..., r_{i_{Q(i,t-1)}}]$ is a $d \times Q(i, t-1)$ matrix whose columns are the instances $r_{i_1}, ..., r_{i_{Q(i,t-1)}}$, and $(l_{i,i_1}, l_{i,i_2}, ..., l_{i,i_{Q(i,t-1)}})^\top$ is a $Q(i, t-1)$-dimensional vector of the corresponding labels observed by node i. The Formula 1 restricts that the weight vector $w_{i,t}$ of the classifier i is only updated on the examples that are positive for its parent node. Then the label vector $\hat{y}_{r_t}$ is computed for the instance r_t, before the real label vector l_{r_t} is observed. Then the current threshold vector θ_t is updated by:

$$\theta_{t+1} = \theta_t + \epsilon(\hat{y}_{r_t} - l_{r_t}), \quad (2)$$

where ϵ is a small positive real number that denotes a corrective step for correcting the current threshold vector θ_t. To illustrate the idea behind the Formula 2, let $y'_t = \hat{y}_{r_t} - l_{r_t}$. Let $y'_{i,t}$ denote an element of the vector y'_t. The Formula 2 correct the current threshold $\theta_{i,t}$ for the classifier i in the following way:

- If $y'_{i,t} = 0$, it means the classifier i made a proper classification for the current instance r_t. Then the current threshold θ_i does not need to be adjusted.

- If $y'_{i,t} = 1$, it means the classifier i made an improper classification by mistakenly identifying the attribute i of the training instance r_t that should have not been identified. This indicates the value of θ_i is not big enough to serve as a threshold so that the attribute i in this case can be filtered out by the classifier i. Therefore, the current threshold θ_i will be adjusted to be larger by ϵ.

- If $y'_{i,t} = -1$, it means the classifier i made an improper classification by failing to identify the attribute i of the training instance r_t that should have been identified. This indicates the value of θ_i is not small enough to serve as a threshold so that the attribute i in this case

Algorithm 1 Hierarchical Learning Algorithm HL-SOT

INITIALIZATION:

Each vector $w_{i,1}, i = 1, ..., N$ of weight matrix W_1 is set to be 0 vector

Threshold vector θ_1 is set to be 0 vector

BEGIN

for $t = 1, ..., |D|$ **do**

  Observe instance $r_t \in \mathcal{X}$

  for $i = 1, ...N$ **do**

    Update each row $w_{i,t}$ of weight matrix W_t by Formula 1

  end for

  Compute $\hat{y}_{r_t} = f(r_t) = g(W_t \cdot r_t)$

  Observe label vector $l_{r_t} \in \mathcal{Y}$ of the instance r_t

  Update threshold vector θ_t by Formula 2

end for

END

can be recognized by the classifier i. Therefore, the current threshold θ_i will be adjusted to be smaller by ϵ.

The hierarchial learning algorithm HL-SOT is presented as in Algorithm 1. The HL-SOT algorithm enables each classifier to have its own specific threshold value and allows this threshold value can be separately learned and corrected through the training process. It is not only a batch-learning setting of the H-RLS algorithm but also a generalization to the latter. If we set the algorithm HL-SOT's parameter ϵ to be 0, the HL-SOT becomes the H-RLS algorithm in a batch-learning setting.

4 Empirical Analysis

In this section, we conduct systematic experiments to perform empirical analysis on our proposed HL-SOT approach against a human-labeled data set. In order to encode each text in the data set by a d-dimensional vector $x \in \mathbb{R}^d$, we first remove all the stop words and then select the top d frequency terms appearing in the data set to construct the index term space. Our experiments are intended to address the following questions:(1) whether utilizing the hierarchical relationships among labels help to improve the accuracy of the classification? (2) whether the introduction of separately learning threshold for each classifier help to improve the accuracy of the classification? (3) how does the corrective step ϵ impact the performance of the proposed approach?(4)how does the dimensionality d of index terms space impact the proposed approach's computing efficiency and accuracy?

4.1 Data Set Preparation

The data set contains 1446 snippets of customer reviews on digital cameras that are collected from a customer review website[4]. We manually construct a SOT for the product of digital cameras. The constructed SOT (e.g., Fig. 1) contains 105 nodes that include 35 non-leaf nodes representing attributes of the digital camera and 70 leaf nodes representing associated sentiments with attribute nodes. Then we label all the snippets with corresponding labels of nodes in the constructed SOT complying with the rule that a target text is to be labeled with a node only if its parent attribute node is labeled with the target text. We randomly divide the labeled data set into five folds so that each fold at least contains one example snippets labeled by each node in the SOT. For each experiment setting, we run 5 experiments to perform cross-fold evaluation by randomly picking three folds as the training set and the other two folds as the testing set. All the testing results are averages over 5 running of experiments.

4.2 Evaluation Metrics

Since the proposed HL-SOT approach is a hierarchical classification process, we use three classic loss functions for measuring classification performance. They are the One-error Loss (O-Loss) function, the Symmetric Loss (S-Loss) function, and the Hierarchical Loss (H-Loss) function:

- One-error loss (O-Loss) function is defined as:
$$L_O(\hat{y}, l) = \mathfrak{B}(\exists i : \hat{y}_i \neq l_i),$$
where $\hat{y}$ is the prediction label vector and l is the true label vector; $\mathfrak{B}$ is the boolean function as defined in Section 3.2.2.

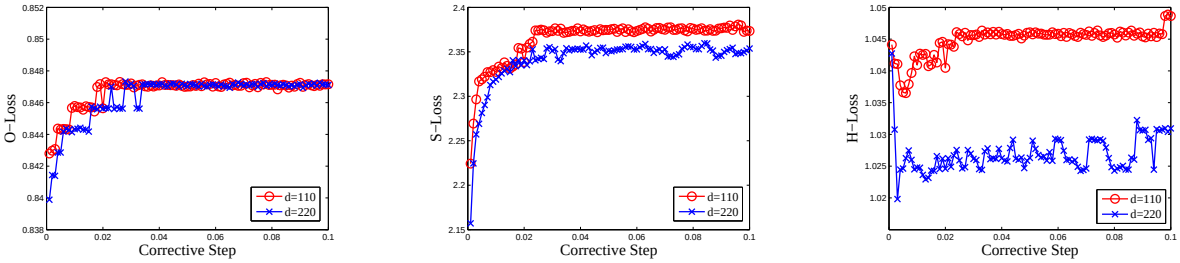
- Symmetric loss (S-Loss) function is defined as:
$$L_S(\hat{y}, l) = \sum_{i=1}^{N} \mathfrak{B}(\hat{y}_i \neq l_i),$$

- Hierarchical loss (H-Loss) function is defined as:
$$L_H(\hat{y}, l) = \sum_{i=1}^{N} \mathfrak{B}(\hat{y}_i \neq l_i \wedge \forall j \in \mathcal{A}(i), \hat{y}_j = l_j),$$

[4] http://www.consumerreview.com/

Table 1: Performance Comparisons (A Smaller Loss Value Means a Better Performance)

Metrics	Dimensinality=110			Dimensinality=220		
	H-RLS	HL-flat	HL-SOT	H-RLS	HL-flat	HL-SOT
O-Loss	0.9812	0.8772	**0.8443**	0.9783	0.8591	**0.8428**
S-Loss	8.5516	2.8921	**2.3190**	7.8623	2.8449	**2.2812**
H-Loss	3.2479	1.1383	**1.0366**	3.1029	1.1298	**1.0247**



(a) O-Loss (b) S-Loss (c) H-Loss

Figure 2: Impact of Corrective Step ϵ

where $\mathcal{A}$ denotes a set of nodes that are ancestors of node i in SOT.

Unlike the O-Loss function and the S-Loss function, the H-Loss function captures the intuition that loss should only be charged on a node whenever a classification mistake is made on a node of SOT but no more should be charged for any additional mistake occurring in the subtree of that node. It measures the discrepancy between the prediction labels and the true labels with consideration on the SOT structure defined over the labels. In our experiments, the recorded loss function values for each experiment running are computed by averaging the loss function values of each testing snippets in the testing set.

4.3 Performance Comparison

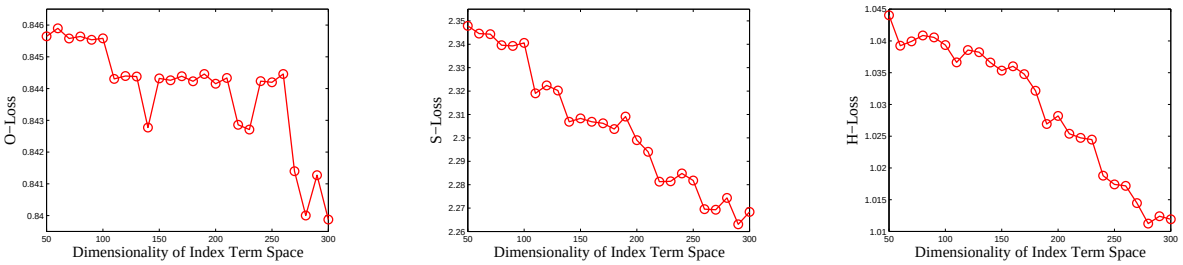
In order to answer the questions (1), (2) in the beginning of this section, we compare our HL-SOT approach with the following two baseline approaches:

- HL-flat: The HL-flat approach involves an algorithm that is a "flat" version of HL-SOT algorithm by ignoring the hierarchical relationships among labels when each classifier is trained. In the training process of HL-flat, the algorithm reflexes the restriction in the HL-SOT algorithm that requires the weight vector $w_{i,t}$ of the classifier i is only updated on the examples that are positive for its parent node.
- H-RLS: The H-RLS approach is implemented by applying the H-RLS algorithm studied in (Cesa-Bianchi et al., 2006). Unlike our proposed HL-SOT algorithm that enables the threshold values to be learned separately for each classifiers in the training process, the H-RLS algorithm only uses an identical threshold values for each classifiers in the classification process.

Experiments are conducted on the performance comparison between the proposed HL-SOT approach with HL-flat approach and the H-RLS approach. The dimensionality d of the index term space is set to be 110 and 220. The corrective step ϵ is set to be 0.005. The experimental results are summarized in Table 1. From Table 1, we can observe that the HL-SOT approach generally beats the H-RLS approach and HL-flat approach on O-Loss, S-Loss, and H-Loss respectively. The H-RLS performs worse than the HL-flat and the HL-SOT, which indicates that the introduction of separately learning threshold for each classifier did improve the accuracy of the classification. The HL-SOT approach performs better than the HL-flat, which demonstrates the effectiveness of utilizing the hierarchical relationships among labels.

4.4 Impact of Corrective Step ϵ

The parameter ϵ in the proposed HL-SOT approach controls the corrective step of the classifiers' thresholds when any mistake is observed in the training process. If the corrective step ϵ is set too large, it might cause the algorithm to be too



(a) O-Loss (b) S-Loss (c) H-Loss

Figure 3: Impact of Dimensionality d of Index Term Space ($\epsilon = 0.005$)

sensitive to each observed mistake. On the contrary, if the corrective step is set too small, it might cause the algorithm not sensitive enough to the observed mistakes. Hence, the corrective step ϵ is a factor that might impact the performance of the proposed approach. Fig. 2 demonstrates the impact of ϵ on O-Loss, S-Loss, and H-Loss. The dimensionality of index term space d is set to be 110 and 220. The value of ϵ is set to vary from 0.001 to 0.1 with each step of 0.001. Fig. 2 shows that the parameter ϵ impacts the classification performance significantly. As the value of ϵ increase, the O-Loss, S-Loss, and H-Loss generally increase (performance decrease). In Fig. 2c it is obviously detected that the H-Loss decreases a little (performance increase) at first before it increases (performance decrease) with further increase of the value of ϵ. This indicates that a finer-grained value of ϵ will not necessarily result in a better performance on the H-loss. However, a fine-grained corrective step generally makes a better performance than a coarse-grained corrective step.

4.5 Impact of Dimensionality d of Index Term Space

In the proposed HL-SOT approach, the dimensionality d of the index term space controls the number of terms to be indexed. If d is set too small, important useful terms will be missed that will limit the performance of the approach. However, if d is set too large, the computing efficiency will be decreased. Fig. 3 shows the impacts of the parameter d respectively on O-Loss, S-Loss, and H-Loss, where d varies from 50 to 300 with each step of 10 and the ϵ is set to be 0.005. From Fig. 3, we observe that as the d increases the O-Loss, S-Loss, and H-Loss generally decrease (performance increase). This means that when more terms are indexed better performance can be achieved by the HL-SOT approach. However,

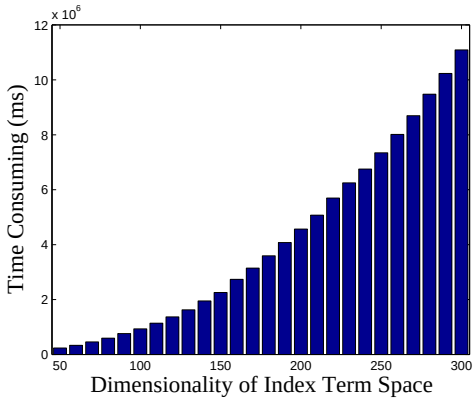


Figure 4: Time Consuming Impacted by d

considering the computing efficiency impacted by d, Fig. 4 shows that the computational complexity of our approach is non-linear increased with d's growing, which indicates that indexing more terms will improve the accuracy of our proposed approach although this is paid by decreasing the computing efficiency.

5 Conclusions, Discussions and Future Work

In this paper, we propose a novel and effective approach to sentiment analysis on product reviews. In our proposed HL-SOT approach, we define SOT to formulate the knowledge of hierarchical relationships among a product's attributes and tackle the problem of sentiment analysis in a hierarchical classification process with the proposed algorithm. The empirical analysis on a human-labeled data set demonstrates the promising results of our proposed approach. The performance comparison shows that the proposed HL-SOT approach outperforms two baselines: the HL-flat and the H-RLS approach. This confirms two intuitive motivations based on which our approach is proposed: 1) separately learning threshold values for

each classifier improve the classification accuracy; 2) knowledge of hierarchical relationships of labels improve the approach's performance. The experiments on analyzing the impact of parameter ϵ indicate that a fine-grained corrective step generally makes a better performance than a coarse-grained corrective step. The experiments on analyzing the impact of the dimensionality d show that indexing more terms will improve the accuracy of our proposed approach while the computing efficiency will be greatly decreased.

The focus of this paper is on analyzing review texts of one product. However, the framework of our proposed approach can be generalized to deal with a mix of review texts of more than one products. In this generalization for sentiment analysis on multiple products reviews, a "big" SOT is constructed and the SOT for each product reviews is a sub-tree of the "big" SOT. The sentiment analysis on multiple products reviews can be performed the same way the HL-SOT approach is applied on single product reviews and can be tackled in a hierarchical classification process with the "big" SOT.

This paper is motivated by the fact that the relationships among a product's attributes could be a useful knowledge for mining product review texts. The SOT is defined to formulate this knowledge in the proposed approach. However, what attributes to be included in a product's SOT and how to structure these attributes in the SOT is an effort of human beings. The sizes and structures of SOTs constructed by different individuals may vary. How the classification performance will be affected by variances of the generated SOTs is worthy of study. In addition, an automatic method to learn a product's attributes and the structure of SOT from existing product review texts will greatly benefit the efficiency of the proposed approach. We plan to investigate on these issues in our future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for many helpful comments on the manuscript. This work is funded by the Research Council of Norway under the VERDIKT research programme (Project No.: 183337).

References

Alina Andreevskaia and Sabine Bergler. 2006. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, Trento, Italy.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research (JMLR)*, 7:31–54.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of 12nd International World Wide Web Conference (WWW'03)*, Budapest, Hungary.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Xiaowen Ding and Bing Liu. 2007. The utility of linguistic rules in opinion mining. In *Proceedings of 30th Annual International ACM Special Interest Group on Information Retrieval Conference (SIGIR'07)*, Amsterdam, The Netherlands.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of 14th ACM Conference on Information and Knowledge Management (CIKM'05)*, Bremen, Germany.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, Madrid, Spain.

Vasileios Hatzivassiloglou and Janyce M. Wiebe. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of 18th International Conference on Computational Linguistics (COLING'00)*, Saarbrüken, Germany.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'04)*, Seattle, USA.

Jaap Kamps, Maarten Marx, R. ort. Mokken, and Maarten de Rijke. 2004. Using WordNet to measure semantic orientation of adjectives. In *Proceedings of 4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of 14th International World Wide Web Conference (WWW'05)*, Chiba, Japan.

Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2007. ARSA: a sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th Annual International ACM Special Interest Group on Information Retrieval Conference (SIGIR'07)*, Amsterdam, The Netherlands.

Yue Lu and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In *Proceedings of 17th International World Wide Web Conference (WWW'08)*, Beijing, China.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of 18th International World Wide Web Conference (WWW'09)*, Madrid, Spain.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP'05)*, Vancouver, Canada.

Ivan Titov and Ryan T. McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of 17th International World Wide Web Conference (WWW'08)*, Beijing, China.

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

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal taxonomies for sentiment analysis. In *Proceedings of 14th ACM Conference on Information and Knowledge Management (CIKM'05)*, Bremen, Germany.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP'05)*, Vancouver, Canada.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of 8th Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*, Sapporo, Japan.

Lina Zhou and Pimwadee Chaovalit. 2008. Ontology-supported polarity mining. *Journal of the American Society for Information Science and Technology (JASIST)*, 59(1):98–110.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and knowledge management (CIKM'06)*, Arlington, USA.

Employing Personal/Impersonal Views in Supervised and Semi-supervised Sentiment Classification

Shoushan Li[†‡] **Chu-Ren Huang**[†] **Guodong Zhou**[‡] **Sophia Yat Mei Lee**[†]

[†]Department of Chinese and Bilingual Studies
The Hong Kong Polytechnic University
{shoushan.li,churenhuang, sophiaym}@gmail.com

[‡] Natural Language Processing Lab
School of Computer Science and Technology
Soochow University, China
gdzhou@suda.edu.cn

Abstract

In this paper, we adopt two views, personal and impersonal views, and systematically employ them in both supervised and semi-supervised sentiment classification. Here, personal views consist of those sentences which directly express speaker's feeling and preference towards a target object while impersonal views focus on statements towards a target object for evaluation. To obtain them, an unsupervised mining approach is proposed. On this basis, an ensemble method and a co-training algorithm are explored to employ the two views in supervised and semi-supervised sentiment classification respectively. Experimental results across eight domains demonstrate the effectiveness of our proposed approach.

1 Introduction

As a special task of text classification, sentiment classification aims to classify a text according to the expressed sentimental polarities of opinions such as '*thumb up*' or '*thumb down*' on the movies (Pang et al., 2002). This task has recently received considerable interests in the Natural Language Processing (NLP) community due to its wide applications.

In general, the objective of sentiment classification can be represented as a kind of binary relation *R*, defined as an ordered triple (*X*, *Y*, *G*), where *X* is an object set including different kinds of people (e.g. writers, reviewers, or users), *Y* is another object set including the target objects (e.g. products, events, or even some people), and *G* is a subset of the Cartesian product $X \times Y$. The concerned relation in sentiment classification is *X* 's evaluation on *Y*, such as '*thumb up*', '*thumb down*', '*favorable*', and '*unfavorable*'. Such relation is usually expressed in text by stating the information involving either a person (one element in *X*) or a target object itself (one element in *Y*). The first type of statement called personal view, e.g. '*I am so happy with this book*', contains *X* 's "subjective" feeling and preference towards a target object, which directly expresses sentimental evaluation. This kind of information is normally domain-independent and serves as highly relevant clues to sentiment classification. The latter type of statement called impersonal view, e.g. '*it is too small*', contains *Y* 's "objective" (i.e. or at least criteria-based) evaluation of the target object. This kind of information tends to contain much domain-specific classification knowledge. Although such information is sometimes not as explicit as personal views in classifying the sentiment of a text, speaker's sentiment is usually implied by the evaluation result.

It is well-known that sentiment classification is very domain-specific (Blitzer et al., 2007), so it is critical to eliminate its dependence on a large-scale labeled data for its wide applications. Since the unlabeled data is ample and easy to collect, a successful semi-supervised sentiment classification system would significantly minimize the involvement of labor and time. Therefore, given the two different views mentioned above, one promising application is to adopt them in co-training algorithms, which has been proven to be an effective semi-supervised learning strategy of incorporating unlabeled data to further improve the classification performance (Zhu, 2005). In addition, we would show that personal/impersonal views are linguistically marked and mining them in text can be easily performed without special annotation.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 414–423,
Uppsala, Sweden, 11-16 July 2010.

In this paper, we systematically employ personal/impersonal views in supervised and semi-supervised sentiment classification. First, an unsupervised bootstrapping method is adopted to automatically separate one document into personal and impersonal views. Then, both views are employed in supervised sentiment classification via an ensemble of individual classifiers generated by each view. Finally, a co-training algorithm is proposed to incorporate unlabeled data for semi-supervised sentiment classification.

The remainder of this paper is organized as follows. Section 2 introduces the related work of sentiment classification. Section 3 presents our unsupervised approach for mining personal and impersonal views. Section 4 and Section 5 propose our supervised and semi-supervised methods on sentiment classification respectively. Experimental results are presented and analyzed in Section 6. Section 7 discusses on the differences between personal/impersonal and subjective/objective. Finally, Section 8 draws our conclusions and outlines the future work.

2 Related Work

Recently, a variety of studies have been reported on sentiment classification at different levels: word level (Esuli and Sebastiani, 2005), phrase level (Wilson et al., 2009), sentence level (Kim and Hovy, 2004; Liu et al., 2005), and document level (Turney, 2002; Pang et al., 2002). This paper focuses on the document-level sentiment classification. Generally, document-level sentiment classification methods can be categorized into three types: unsupervised, supervised, and semi-supervised.

Unsupervised methods involve deriving a sentiment classifier without any labeled documents. Most of previous work use a set of labeled sentiment words called seed words to perform unsupervised classification. Turney (2002) determines the sentiment orientation of a document by calculating point-wise mutual information between the words in the document and the seed words of '*excellent*' and '*poor*'. Kennedy and Inkpen (2006) use a term-counting method with a set of seed words to determine the sentiment. Zagibalov and Carroll (2008) first propose a seed word selection approach and then apply the same term-counting method for Chinese sentiment classifications. These unsupervised approaches are believed to be domain-independent for sentiment classification.

Supervised methods consider sentiment classification as a standard classification problem in which labeled data in a domain are used to train a domain-specific classifier. Pang et al. (2002) are the first to apply supervised machine learning methods to sentiment classification. Subsequently, many other studies make efforts to improve the performance of machine learning-based classifiers by various means, such as using subjectivity summarization (Pang and Lee, 2004), seeking new superior textual features (Riloff et al., 2006), and employing document subcomponent information (McDonald et al., 2007). As far as the challenge of domain-dependency is concerned, Blitzer et al. (2007) present a domain adaptation approach for sentiment classification.

Semi-supervised methods combine unlabeled data with labeled training data (often small-scaled) to improve the models. Compared to the supervised and unsupervised methods, semi-supervised methods for sentiment classification are relatively new and have much less related studies. Dasgupta and Ng (2009) integrate various methods in semi-supervised sentiment classification including spectral clustering, active learning, transductive learning, and ensemble learning. They achieve a very impressive improvement across five domains. Wan (2009) applies a co-training method to semi-supervised learning with labeled English corpus and unlabeled Chinese corpus for Chinese sentiment classification.

3 Unsupervised Mining of Personal and Impersonal Views

As mentioned in Section 1, the objective of sentiment classification is to classify a specific binary relation: *X* 's evaluation on *Y*, where *X* is an object set including different kinds of persons and *Y* is another object set including the target objects to be evaluated. First of all, we focus on an analysis on sentences in product reviews regarding the two views: personal and impersonal views.

The personal view consists of personal sentences (i.e. *X* 's sentences) exemplified below:

I. Personal preference:
   E1: *I love this breadmaker!*
   E2: *I disliked it from the beginning.*
II. Personal emotion description:
   E3: *Very disappointed!*
   E4: *I am happy with the product.*
III. Personal actions:

E5: *Do not waste your money.*

E6: *I have recommended this machine to all my friends.*

The impersonal view consists of impersonal sentences (i.e. Y 's sentences) exemplified below:

I. Impersonal feature description:

E7: *They are too thin to start with.*

E8: *This product is extremely quiet.*

II. Impersonal evaluation:

E9: *It's great.*

E10: *The product is a waste of time and money.*

III. Impersonal actions:

E11: *This product not even worth a penny.*

E12: *It broke down again and again.*

We find that the subject of a sentence presents important cues for personal/impersonal views, even though a formal and computable definition of this contrast cannot be found. Here, subject refers to one of the two main constituents in the traditional English grammar (the other constituent being the predicate) (Crystal, 2003)[1]. For example, the subjects in the above examples of E1, E7 and E11 are '*I*', '*they*', and '*this product*' respectively. For automatic mining the two views, personal/impersonal sentences can be defined according to their subjects:

Personal sentence: the sentence whose subject is (or represents) a person.

Impersonal sentence: the sentence whose subject is not (does not represent) a person.

In this study, we mainly focus on product review classification where the target object in the set Y is not a person. The definitions need to be adjusted when the evaluation target itself is a person, e.g. the political sentiment classification by Durant and Smith (2007).

Our unsupervised mining approach for mining personal and impersonal sentences consists of two main steps. First, we extract an initial set of personal and impersonal sentences with some heuristic rules: If the first word of one sentence is (or implies) a personal pronoun including '*I*', '*we*', and '*do*', then the sentence is extracted as a personal sentence; If the first word of one sentence is an impersonal pronoun including '*it*', '*they*', '*this*', and '*these*', then the sentence is extracted as an impersonal sentence. Second, we apply the classifier which is trained with the initial set of personal and impersonal sentences to classify the remaining sentences. This step aims to classify the sentences without pronouns (e.g. E3). Figure 1 shows the unsupervised mining algorithm.

Input:

The training data D

Output:

All personal and impersonal sentences, i.e. sentence sets $S_{personal}$ and $S_{impersonal}$.

Procedure:

(1). Segment all documents in D to sentences S using punctuations (such as periods and interrogation marks)

(2). Apply the heuristic rules to classify the sentences S with proper pronouns into, S_{p1} and S_{i1}

(3). Train a binary classifier f_{p-i} with S_{p1} and S_{i1}

(4). Use f_{p-i} to classify the remaining sentences into S_{p2} and S_{i2}

(5). $S_{personal} = S_{p1} \bigcup S_{p2}$, $S_{impersonal} = S_{i1} \bigcup S_{i2}$

Figure 1: The algorithm for unsupervised mining personal and impersonal sentences from a training data

4 Employing Personal/Impersonal Views in Supervised Sentiment Classification

After unsupervised mining of personal and impersonal sentences, the training data is divided into two views: the personal view, which contains personal sentences, and the impersonal view, which contains impersonal sentences. Obviously, these two views can be used to train two different classifiers, f_1 and f_2, for sentiment classification respectively.

Since our mining approach is unsupervised, there inevitably exist some noises. In addition, the sentences of different views may share the same information for sentiment classification. For example, consider the following two sentences: '*It is a waste of money*.' and '*Do not waste your money*.' Apparently, the first one belongs to the impersonal view while the second one belongs to personal view, according to our heuristic rules. However, these two sentences share the same word, '*waste*', which conveys strong negative sentiment information. This suggests that training a single-view classifier f_3 with all sentences should help. Therefore, three base classifiers, f_1, f_2, and f_3, are eventually derived from the personal view, the impersonal

[1] The subject has the grammatical function in a sentence of relating its constituent (a noun phrase) by means of the verb to any other elements present in the sentence, i.e. objects, complements, and adverbials.

view and the single view, respectively. Each base classifier provides not only the class label outputs but also some kinds of confidence measurements, e.g. posterior probabilities of the testing sample belonging to each class.

Formally, each base classifier f_l $(l = 1,2,3)$ assigns a test sample (denoted as x_l) a posterior probability vector $\vec{P}(x_l)$:

$$\vec{P}(x_l) = < p(c_1 \mid x_l), p(c_2 \mid x_l) >^t$$

where $p(c_1 \mid x_l)$ denotes the probability that the l-th base classifier considers the sample belonging to c_1.

In the ensemble learning literature, various methods have been presented for combining base classifiers. The combining methods are categorized into two groups (Duin, 2002): fixed rules such as voting rule, product rule, and sum rule (Kittler et al., 1998), and trained rules such as weighted sum rule (Fumera and Roli, 2005) and meta-learning approaches (Vilalta and Drissi, 2002). In this study, we choose a fixed rule and a trained rule to combine the three base classifiers f_1, f_2, and f_3.

The chosen fixed rule is product rule which combine base classifiers by multiplying the posterior possibilities and using the multiplied possibility for decision, i.e.

$$assign \quad y \to c_j$$

$$where \ \ j = \arg\max_i \prod_{l=1}^{3} p(c_i \mid x_l)$$

The chosen trained rule is stacking (Vilalta and Drissi, 2002; Džeroski and Ženko, 2004) where a meta-classifier is trained with the output of the base classifiers as the input. Formally, let x' denote a feature vector of a sample from the development data. The output of the l-th base classifier f_l on this sample is the probability distribution over the category set $\{c_1, c_2\}$, i.e.

$$\vec{P}(x'_l) = < p(c_1 \mid x'_l), p_l(c_2 \mid x'_l) >$$

Then, a meta-classifier is trained using the development data with the meta-level feature vector $x^{meta} \in R^{2\times 3}$

$$x^{meta} = < \vec{P}(x'_{l=1}), \vec{P}(x'_{l=2}), \vec{P}(x'_{l=3}) >$$

In our experiments, we perform stacking with 4-fold cross validation to generate meta-training data where each fold is used as the development data and the other three folds are used to train the base classifiers in the training phase.

5 Employing Personal/Impersonal Views in Semi-Supervised Sentiment Classification

Semi-supervised learning is a strategy which combines unlabeled data with labeled training data to improve the models. Given the two-view classifiers f_1 and f_2 along with the single-view classifier f_3, we perform a co-training algorithm for semi-supervised sentiment classification. The co-training algorithm is a specific semi-supervised learning approach which starts with a set of labeled data and increases the amount of labeled data using the unlabeled data by bootstrapping (Blum and Mitchell, 1998). Figure 2 shows the co-training algorithm in our semi-supervised sentiment classification.

Input:

The labeled data L containing personal sentence set $S_{L-personal}$ and impersonal sentence set $S_{L-impersonal}$

The unlabeled data U containing personal sentence set $S_{U-personal}$ and impersonal sentence set $S_{U-impersonal}$

Output:

New labeled data L

Procedure:

Loop for N iterations until $U = \phi$

(1). Learn the first classifier f_1 *with* $S_{L-personal}$

(2). Use f_1 *to label samples from U with* $S_{U-personal}$

(3). Choose n_1 *positive and* n_1 *negative most confidently predicted samples* A_1

(4). Learn the second classifier f_2 *with* $S_{L-impersonal}$

(5). Use f_2 *to label samples from U with* $S_{U-impersonal}$

(6). Choose n_2 *positive and* n_2 *negative most confidently predicted samples* A_2

(7). Learn the third classifier f_3 *with* L

(8). Use f_3 *to label samples from U*

(9). Choose n_3 *positive and* n_3 *negative most confidently predicted samples* A_3

(10). Add samples $A_1 \cup A_2 \cup A_3$ *with the corresponding labels into* L

(11). Update $S_{L-personal}$ *and* $S_{L-impersonal}$

Figure 2: Our co-training algorithm for semi-supervised sentiment classification

After obtaining the new labeled data, we can either adopt one classifier (i.e. f_3) or a combined classifier (i.e. $f_1 + f_2 + f_3$) in further training and testing. In our experimentation, we explore both of them with the former referred to as **co-training and single classifier** and the latter referred to as **co-training and combined classifier**.

6 Experimental Studies

We have systematically explored our method on product reviews from eight domains: book, DVD, electronic appliances, kitchen appliances, health, network, pet and software.

6.1 Experimental Setting

The product reviews on the first four domains (book, DVD, electronic, and kitchen appliances) come from the multi-domain sentiment classification corpus, collected from http://www.amazon.com/ by Blitzer et al. (2007)[2]. Besides, we also collect the product views from http://www.amazon.com/ on other four domains (health, network, pet and software)[3]. Each of the eight domains contains 1000 positive and 1000 negative reviews. Figure 3 gives the distribution of personal and impersonal sentences in the training data (75% labeled data of all data). It shows that there are more impersonal sentences than personal ones in each domain, in particular in the DVD domain, where the number of impersonal sentences is at least twice as many as that of personal sentences. This unusual phenomenon is mainly attributed to the fact that many objective descriptions, e.g. the movie plot introductions, are expressed in the DVD domain which makes the extracted personal and impersonal sentences rather unbalanced.

We apply both support vector machine (SVM) and Maximum Entropy (ME) algorithms with the help of the SVM-light[4] and Mallet[5] tools. All parameters are set to their default values. We find that ME performs slightly better than SVM on the average. Furthermore, ME offers posterior probability information which is required for combination methods. Thus we apply the ME classification algorithm for further combination and co-training. In particular, we only employ Boolean features, representing the presence or absence of a word in a document. Finally, we perform *t*-test to evaluate the significance of the performance difference between two systems with different methods (Yang and Liu, 1999).

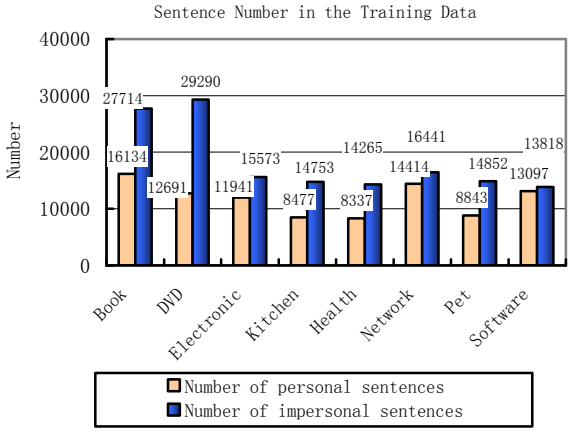


Figure 3: Distribution of personal and impersonal sentences in the training data of each domain

6.2 Experimental Results on Supervised Sentiment Classification

4-fold cross validation is performed for supervised sentiment classification. For comparison, we generate two random views by randomly splitting the whole feature space into two parts. Each part is seen as a view and used to train a classifier. The combination (two random view classifiers along with the single-view classifier f_3) results are shown in the last column of Table 1. The comparison between random two views and our proposed two views will clarify whether the performance gain comes truly from our proposed two-view mining, or simply from using the classifier combination strategy.

Table 1 shows the performances of different classifiers, where the single-view classifier f_3 which uses all sentences for training and testing, is considered as our baseline. Note that the baseline performances of the first four domains are worse than the ones reported in Blitzer et al. (2007). But their experiment is performed with only one split on the data with 80% as the training data and 20% as the testing data, which means the size of their training data is larger than ours. Also, we find that our performances are similar to the ones (described as fully supervised results) reported in Dasgupta and Ng (2009) where the same data in the four domains are used and 10-fold cross validation is performed.

[2] http://www.seas.upenn.edu/~mdredze/datasets/sentiment/

[3] Note that the second version of multi-domain sentiment classification corpus does contain data from many other domains. However, we find that the reviews in the other domains contain many duplicated samples. Therefore, we re-collect the reviews from http://www.amazon.com/ and filter those duplicated ones. The new collection is here: http://llt.cbs.polyu.edu.hk/~lss/ACL2010_Data_SSLi.zip

[4] http://svmlight.joachims.org/

[5] http://mallet.cs.umass.edu/

Domain	Personal View Classifier f_1	Impersonal View Classifier f_2	Single View Classifier (**baseline**) f_3	Combination (Stacking) $f_1 + f_2 + f_3$	Combination (Product rule) $f_1 + f_2 + f_3$	Combination with two random views (Product rule)
Book	0.7004	0.7474	0.7654	0.7919	**0.7949**	0.7546
DVD	0.6931	0.7663	0.7884	0.8079	**0.8165**	0.8054
Electronic	0.7414	0.7844	0.8074	0.8304	**0.8364**	0.8210
Kitchen	0.7430	0.8030	0.8290	0.8555	**0.8565**	0.8152
Health	0.7000	0.7370	0.7559	0.7780	**0.7815**	0.7548
Network	0.7655	0.7710	0.8265	0.8360	**0.8435**	0.8312
Pet	0.6940	0.7145	0.7390	0.7565	**0.7665**	0.7423
Software	0.7035	0.7205	0.7470	**0.7730**	0.7715	0.7615
AVERAGE	*0.7176*	*0.7555*	*0.7823*	*0.8037*	***0.8084***	*0.7858*

Table 1: Performance of supervised sentiment classification

From Table 1, we can see that impersonal view classifier f_1 consistently performs better than personal view classifier f_2. Similar to the sentence distributions, the difference in the classification performances between these two views in the DVD domain is the largest (0.6931 vs. 0.7663).

Both the combination methods (stacking and product rule) significantly outperform the baseline in each domain (p-value<0.01) with a decent average performance improvement of 2.61%. Although the performance difference between the product rule and stacking is not significant, the product rule is obviously a better choice as it involves much easier implementation. Therefore, in the semi-supervised learning process, we only use the product rule to combine the individual classifiers. Finally, it shows that random generation of two views with the combination method of the product rule only slightly outperforms the baseline on the average (*0.7858* vs. *0.7823*) but performs much worse than our unsupervised mining of personal and impersonal views.

6.3 Experimental Results on Semi-supervised Sentiment Classification

We systematically evaluate and compare our two-view learning method with various semi-supervised ones as follows:

Self-training, which uses the unlabeled data in a bootstrapping way like co-training yet limits the number of classifiers and the number of views to one. Only the baseline classifier f_3 is used to select most confident unlabeled samples in each iteration.

Transductive SVM, which seeks the largest separation between labeled and unlabeled data through regularization (Joachims, 1999). We implement it with the help of the SVM-light tool.

Co-training with random two-view generation (briefly called **co-training with random views**), where two views are generated by randomly splitting the whole feature space into two parts.

In semi-supervised sentiment classification, the data are randomly partitioned into labeled training data, unlabeled data, and testing data with the proportion of 10%, 70% and 20% respectively. Figure 4 reports the classification accuracies in all iterations, where **baseline** indicates the supervised classifier f_3 trained on the 10% data; both **co-training and single classifier** and **co-training and combined classifier** refer to co-training using our proposed personal and impersonal views. But the former merely applies the baseline classifier f_3 trained the new labeled data to test on the testing data while the latter applies the combined classifier $f_1 + f_2 + f_3$. In each iteration, two top-confident samples in each category are chosen, i.e. $n_1 = n_2 = n_3 = 2$. For clarity, results of other methods (e.g. **self-training**, **transductive SVM**) are not shown in Figure 4 but will be reported in Figure 5 later.

Figure 4 shows that **co-training and combined classifier** always outperforms **co-training and single classifier**. This again justifies the effectiveness of our two-view learning on supervised sentiment classification.

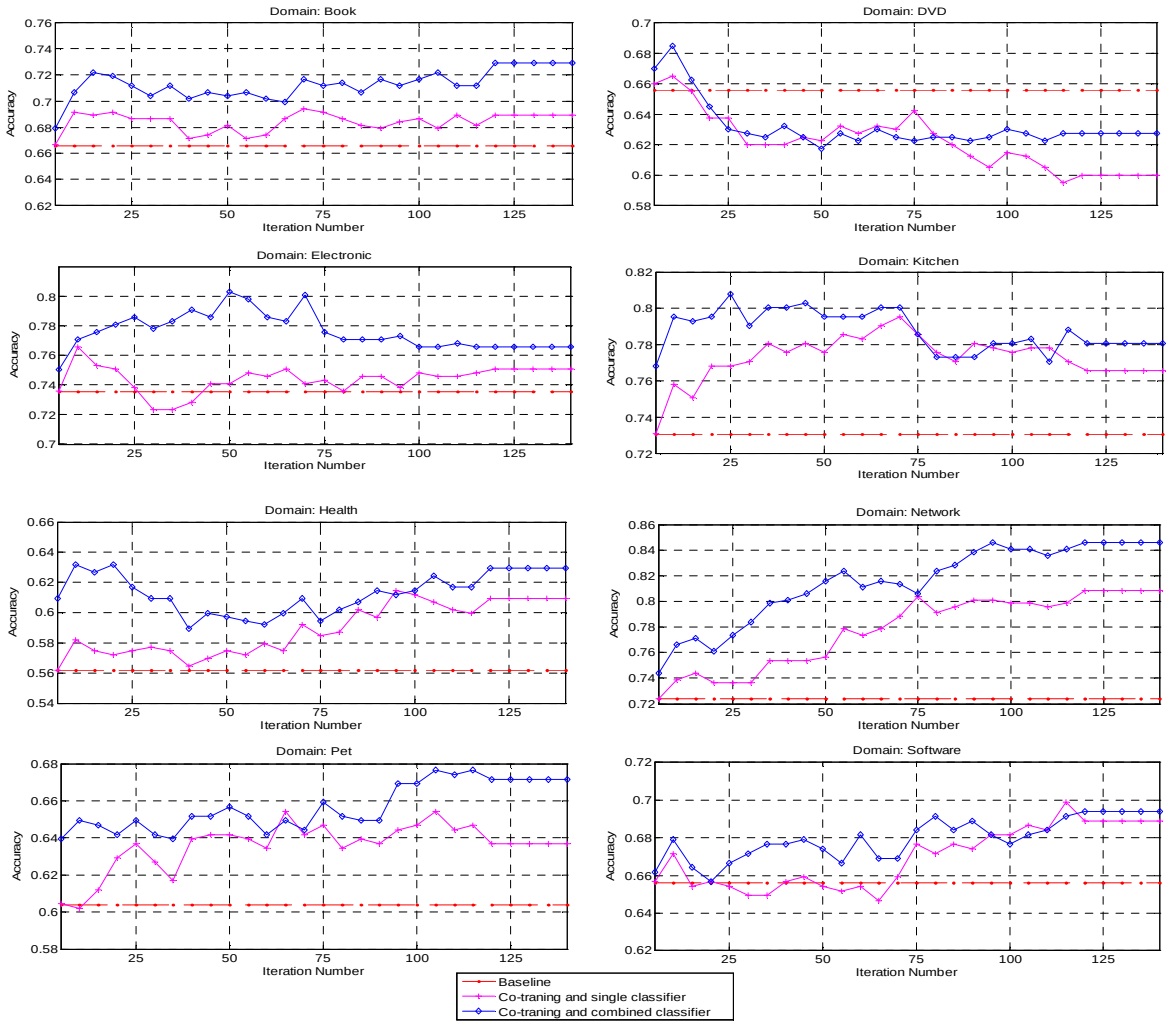


Figure 4: Classification performance vs. iteration numbers (using 10% labeled data as training data)

One open question is whether the unlabeled data improve the performance. Let us set aside the influence of the combination strategy and focus on the effectiveness of semi-supervised learning by comparing the baseline and **co-training and single classifier**. Figure 4 shows different results on different domains. Semi-supervised learning fails on the DVD domain while on the three domains of book, electronic, and software, semi-supervised learning benefits slightly (p-value>0.05). In contrast, semi-supervised learning benefits much on the other four domains (health, kitchen, network, and pet) from using unlabeled data and the performance improvements are statistically significant (p-value<0.01). Overall speaking, we think that the unlabeled data are very helpful as they lead to about 4% accuracy improvement on the average except for the DVD domain. Along with the supervised combination strategy, our approach can significantly improve the performance more than 7% on the average compared to the baseline.

Figure 5 shows the classification results of different methods with different sizes of the labeled data: 5%, 10%, and 15% of all data, where the testing data are kept the same (20% of all data). Specifically, the results of other methods including **self-training**, **transductive SVM**, and **random views** are presented when 10% labeled data are used in training. It shows that **self-training** performs much worse than our approach and fails to improve the performance of five of the eight domains. **Transductive SVM** performs even worse and can only improve the performance of the "software" domain. Although c**o-training with random views** outperforms the baseline on four of the eight domains, it performs worse than **co-training and single classifier.** This suggests that the impressive improvements are mainly due to our unsupervised two-view mining rather than the combination strategy.

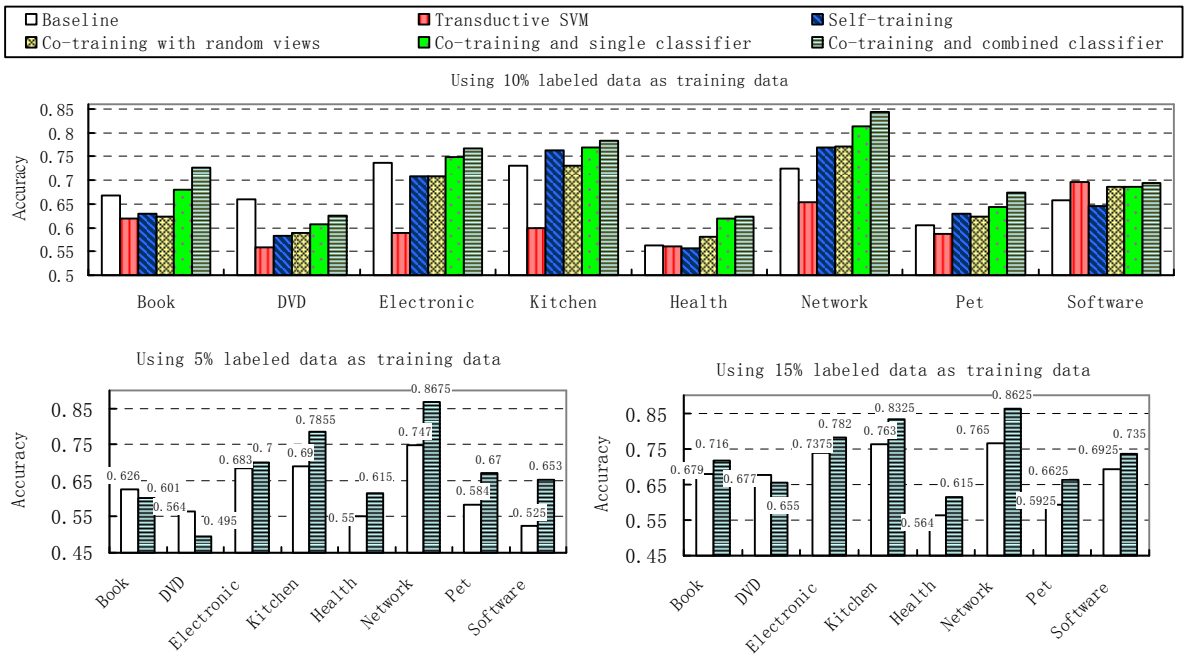


Figure 5: Performance of semi-supervised sentiment classification when 5%, 10%, and 15% labeled data are used

Figure 5 also shows that our approach is rather robust and achieves excellent performances in different training data sizes, although our approach fails on two domains, i.e. book and DVD, when only 5% of the labeled data are used. This failure may be due to that some of the samples in these two domains are too ambiguous and hard to classify. Manual checking shows that quite a lot of samples on these two domains are even too difficult for professionals to give a high-confident label. Another possible reason is that there exist too many objective descriptions in these two domains, thus introducing too much noisy information for semi-supervised learning.

The effectiveness of different sizes of chosen samples in each iteration is also evaluated like $n_1 = n_2 = n_3 = 6$ and $n_1 = 3, n_2 = n_3 = 6$ (This assignment is considered because the personal view classifier performs worse than the other two classifiers). Our experimental results are still unsuccessful in the DVD domain and do not show much difference on other domains. We also test the co-training approach without the single-view classifier f_3. Experimental results show that the inclusion of the single-view classifier f_3 slightly helps the co-training approach. The detailed discussion of the results is omitted due to space limit.

6.4 Why our approach is effective?

One main reason for the effectiveness of our approach on supervised learning is the way how personal and impersonal views are dealt with. As personal and impersonal views have different ways of expressing opinions, splitting them into two separations can filter some classification noises. For example, in the sentence of "*I have seen amazing dancing, and good dancing. This was TERRIBLE dancing!*". The first sentence is classified as a personal sentence and the second one is an impersonal sentence. Although the words '*amazing*' and '*good*' convey strong positive sentiment information, the whole text is negative. If we get the bag-of-words from the whole text, the classification result will be wrong. Rather, splitting the text into two parts based on different views allows correct classification as the personal view rarely contains impersonal words such as '*amazing*' and '*good*'. The classification result will thus be influenced by the impersonal view.

In addition, a document may contain both personal and impersonal sentences, and each of them, to a certain extent, , provides classification evidence. In fact, we randomly select 50 documents in the domain of kitchen appliances and find that 80% of the documents take both personal and impersonal sentences in which both of them express explicit opinions. That is to say, the two views provide different, complementary information for classification. This qualifies the success requirement of co-training algorithm to some extend. This might be the reason for the effectiveness of our approach on semi-supervised learning.

7 Discussion on Personal/Impersonal vs. Subjective/Objective

As mentioned in Section 1, personal view contains *X* 's "subjective" feeling, and impersonal view contains *Y* 's "objective" (i.e. or at least criteria-based) evaluation of the target object. However, our technically-defined concepts of personal/impersonal are definitely different from subjective/objective: Personal view can certainly contain many objective expressions, e.g. '*I bought this electric kettle*' and impersonal view can contain many subjective expressions, e.g. '*It is disappointing*'.

Our technically-defined personal/impersonal views are two different ways to describe opinions. Personal sentences are often used to express opinions in a direct way and their target object should be one of *X*. Impersonal ones are often used to express opinions in an indirect way and their target object should be one of *Y*. The ideal definition of personal (or impersonal) view given in Section 1 is believed to be a subset of our technical definition of personal (or impersonal) view. Thus impersonal view may contain both *Y* 's objective evaluation (more likely to be domain independent) and subjective *Y*'s description.

In addition, simply splitting text into subjective/objective views is not particularly helpful. Since a piece of objective text provides rather limited implicit classification information, the classification abilities of the two views are very unbalanced. This makes the co-training process unfeasible. Therefore, we believe that our technically-defined personal/impersonal views are more suitable for two-view learning compared to subjective/objective views.

8 Conclusion and Future Work

In this paper, we propose a robust and effective two-view model for sentiment classification based on personal/impersonal views. Here, the personal view consists of subjective sentences whose subject is a person, whereas the impersonal view consists of objective sentences whose subject is not a person. Such views are lexically cued and can be obtained without pre-labeled data and thus we explore an unsupervised learning approach to mine them. Combination methods and a co-training algorithm are proposed to deal with supervised and semi-supervised sentiment classification respectively. Evaluation on product reviews from eight domains shows that our approach significantly improves the performance across all eight domains on supervised sentiment classification and greatly outperforms the baseline with more than 7% accuracy improvement on the average across seven of eight domains (except the DVD domain) on semi-supervised sentiment classification.

In the future work, we will integrate the subjectivity summarization strategy (Pang and Lee, 2004) to help discard noisy objective sentences. Moreover, we need to consider the cases when both *X* and *Y* appear in a sentence. For example, the sentence "*I think they're poor*" should be an impersonal view but wrongly classified as a personal one according to our technical rules. We believe that these will help improve our approach and hopefully are applicable to the DVD domain. Another interesting and practical idea is to integrate active learning (Settles, 2009), another popular but principally different kind of semi-supervised learning approach, with our two-view learning approach to build high-performance systems with the least labeled data.

Acknowledgments

The research work described in this paper has been partially supported by Start-up Grant for Newly Appointed Professors, No. 1-BBZM in the Hong Kong Polytechnic University and two NSFC grants, No. 60873150 and No. 90920004. We also thank the three anonymous reviewers for their invaluable comments.

References

Blitzer J., M. Dredze, and F. Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of ACL-07*.

Blum A. and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT-98*.

Crystal D. 2003. The Cambridge Encyclopedia of the English Language. Cambridge University Press.

Dasgupta S. and V. Ng. 2009. Mine the Easy and Classify the Hard: Experiments with Automatic Sentiment Classification. In *Proceedings of ACL-IJCNLP-09*.

Duin R. 2002. The Combining Classifier: To Train Or Not To Train? In *Proceedings of 16th International Conference on Pattern Recognition (ICPR-02)*.

Durant K. and M. Smith. 2007. Predicting the Political Sentiment of Web Log Posts using

Supervised Machine Learning Techniques Coupled with Feature Selection. In Processing of *Advances in Web Mining and Web Usage Analysis*.

Džeroski S. and B. Ženko. 2004. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning*, vol.54(3), pp.255-273, 2004.

Esuli A. and F. Sebastiani. 2005. Determining the Semantic Orientation of Terms through Gloss Classification. In *Proceedings of CIKM-05*.

Fumera G. and F. Roli. 2005. A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems. *IEEE Trans. PAMI*, vol.27, pp.942–956, 2005

Joachims, T. 1999. Transductive Inference for Text Classification using Support Vector Machines. ICML1999.

Kennedy A. and D. Inkpen. 2006. Sentiment Classification of Movie Reviews using Contextual Valence Shifters. *Computational Intelligence*, vol.22(2), pp.110-125, 2006.

Kim S. and E. Hovy. 2004. Determining the Sentiment of Opinions. In *Proceedings of COLING-04*.

Kittler J., M. Hatef, R. Duin, and J. Matas. 1998. On Combining Classifiers. *IEEE Trans. PAMI*, vol.20, pp.226-239, 1998

Liu B., M. Hu, and J. Cheng. 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of WWW-05*.

McDonald R., K. Hannan, T. Neylon, M. Wells, and J. Reynar. 2007. Structured Models for Fine-to-coarse Sentiment Analysis. In *Proceedings of ACL-07*.

Pang B. and L. Lee. 2004. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In *Proceedings of ACL-04*.

Pang B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of EMNLP-02*.

Riloff E., S. Patwardhan, and J. Wiebe. 2006. Feature Subsumption for Opinion Analysis. In *Proceedings of EMNLP-06*.

Settles B. 2009. Active Learning Literature Survey. *Technical Report 1648*, Department of Computer Sciences, University of Wisconsin at Madison, Wisconsin.

Turney P. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of ACL-02*.

Vilalta R. and Y. Drissi. 2002. A Perspective View and Survey of Meta-learning. *Artificial Intelligence Review*, 18(2): 77–95.

Wan X. 2009. Co-Training for Cross-Lingual Sentiment Classification. In *Proceedings of ACL-IJCNLP-09*.

Wilson T., J. Wiebe, and P. Hoffmann. 2009. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, vol.35(3), pp.399-433, 2009.

Yang Y. and X. Liu. 1999. A Re-Examination of Text Categorization methods. In *Proceedings of SIGIR-99*.

Zagibalov T. and J. Carroll. 2008. Automatic Seed Word Selection for Unsupervised Sentiment Classification of Chinese Test. In *Proceedings of COLING-08*.

Zhu X. 2005. Semi-supervised Learning Literature Survey. *Technical Report Computer Sciences 1530*, University of Wisconsin – Madison.

A Latent Dirichlet Allocation method for Selectional Preferences

Alan Ritter, Mausam and Oren Etzioni
Department of Computer Science and Engineering
Box 352350, University of Washington, Seattle, WA 98195, USA
{*aritter,mausam,etzioni*}*@cs.washington.edu*

Abstract

The computation of *selectional preferences*, the admissible argument values for a relation, is a well-known NLP task with broad applicability. We present LDA-SP, which utilizes LinkLDA (Erosheva et al., 2004) to model selectional preferences. By simultaneously inferring latent topics and topic distributions over relations, LDA-SP combines the benefits of previous approaches: like traditional class-based approaches, it produces human-interpretable classes describing each relation's preferences, but it is competitive with non-class-based methods in predictive power.

We compare LDA-SP to several state-of-the-art methods achieving an 85% increase in recall at 0.9 precision over mutual information (Erk, 2007). We also evaluate LDA-SP's effectiveness at filtering improper applications of inference rules, where we show substantial improvement over Pantel *et al.*'s system (Pantel et al., 2007).

1 Introduction

Selectional Preferences encode the set of admissible argument values for a relation. For example, locations are likely to appear in the second argument of the relation *X is headquartered in Y* and companies or organizations in the first. A large, high-quality database of preferences has the potential to improve the performance of a wide range of NLP tasks including semantic role labeling (Gildea and Jurafsky, 2002), pronoun resolution (Bergsma et al., 2008), textual inference (Pantel et al., 2007), word-sense disambiguation (Resnik, 1997), and many more. Therefore, much attention has been focused on automatically computing them based on a corpus of relation instances.

Resnik (1996) presented the earliest work in this area, describing an information-theoretic approach that inferred selectional preferences based on the WordNet hypernym hierarchy. Recent work (Erk, 2007; Bergsma et al., 2008) has moved away from generalization to known classes, instead utilizing distributional similarity between nouns to generalize beyond observed relation-argument pairs. This avoids problems like WordNet's poor coverage of proper nouns and is shown to improve performance. These methods, however, no longer produce the generalized class for an argument.

In this paper we describe a novel approach to computing selectional preferences by making use of unsupervised topic models. Our approach is able to combine benefits of both kinds of methods: it retains the generalization and human-interpretability of class-based approaches and is also competitive with the direct methods on predictive tasks.

Unsupervised topic models, such as latent Dirichlet allocation (LDA) (Blei et al., 2003) and its variants are characterized by a set of hidden topics, which represent the underlying semantic structure of a document collection. For our problem these topics offer an intuitive interpretation – they represent the (latent) set of classes that store the preferences for the different relations. Thus, topic models are a natural fit for modeling our relation data.

In particular, our system, called LDA-SP, uses LinkLDA (Erosheva et al., 2004), an extension of LDA that simultaneously models *two* sets of distributions for each topic. These two sets represent the two arguments for the relations. Thus, LDA-SP is able to capture information about the pairs of topics that commonly co-occur. This information is very helpful in guiding inference.

We run LDA-SP to compute preferences on a massive dataset of binary relations $r(a_1, a_2)$ ex-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 424–434,
Uppsala, Sweden, 11-16 July 2010.

tracted from the Web by TEXTRUNNER (Banko and Etzioni, 2008). Our experiments demonstrate that LDA-SP significantly outperforms state of the art approaches obtaining an 85% increase in recall at precision 0.9 on the standard pseudo-disambiguation task.

Additionally, because LDA-SP is based on a formal probabilistic model, it has the advantage that it can naturally be applied in many scenarios. For example, we can obtain a better understanding of similar relations (Table 1), filter out incorrect inferences based on querying our model (Section 4.3), as well as produce a repository of class-based preferences with a little manual effort as demonstrated in Section 4.4. In all these cases we obtain high quality results, for example, massively outperforming Pantel et al.'s approach in the textual inference task.[1]

2 Previous Work

Previous work on selectional preferences can be broken into four categories: class-based approaches (Resnik, 1996; Li and Abe, 1998; Clark and Weir, 2002; Pantel et al., 2007), similarity based approaches (Dagan et al., 1999; Erk, 2007), discriminative (Bergsma et al., 2008), and generative probabilistic models (Rooth et al., 1999).

Class-based approaches, first proposed by Resnik (1996), are the most studied of the four. They make use of a pre-defined set of classes, either manually produced (e.g. WordNet), or automatically generated (Pantel, 2003). For each relation, some measure of the overlap between the classes and observed arguments is used to identify those that best describe the arguments. These techniques produce a human-interpretable output, but often suffer in quality due to an incoherent taxonomy, inability to map arguments to a class (poor lexical coverage), and word sense ambiguity.

Because of these limitations researchers have investigated non-class based approaches, which attempt to directly classify a given noun-phrase as plausible/implausible for a relation. Of these, the *similarity based approaches* make use of a distributional similarity measure between arguments and evaluate a heuristic scoring function:

$$S_{\text{rel}}(\text{arg}) = \sum_{\text{arg}' \in \text{Seen}(\text{rel})} \text{sim}(\text{arg}, \text{arg}') \cdot \text{wt}_{\text{rel}}(\text{arg})$$

[1] Our repository of selectional preferences is available at `http://www.cs.washington.edu/research/ldasp`.

Erk (2007) showed the advantages of this approach over Resnik's information-theoretic class-based method on a pseudo-disambiguation evaluation. These methods obtain better lexical coverage, but are unable to obtain any abstract representation of selectional preferences.

Our solution fits into the general category of *generative probabilistic models*, which model each relation/argument combination as being generated by a latent class variable. These classes are automatically learned from the data. This retains the class-based flavor of the problem, without the knowledge limitations of the explicit class-based approaches. Probably the closest to our work is a model proposed by Rooth et al. (1999), in which each class corresponds to a multinomial over relations and arguments and EM is used to learn the parameters of the model. In contrast, we use a LinkLDA framework in which each relation is associated with a corresponding multinomial distribution over classes, and each argument is drawn from a class-specific distribution over words; LinkLDA captures co-occurrence of classes in the two arguments. Additionally we perform full Bayesian inference using collapsed Gibbs sampling, in which parameters are integrated out (Griffiths and Steyvers, 2004).

Recently, Bergsma *et. al.* (2008) proposed the first *discriminative approach* to selectional preferences. Their insight that pseudo-negative examples could be used as training data allows the application of an SVM classifier, which makes use of many features in addition to the relation-argument co-occurrence frequencies used by other methods. They automatically generated positive and negative examples by selecting arguments having high and low mutual information with the relation. Since it is a discriminative approach it is amenable to feature engineering, but needs to be retrained and tuned for each task. On the other hand, generative models produce complete probability distributions of the data, and hence can be integrated with other systems and tasks in a more principled manner (see Sections 4.2.2 and 4.3.1). Additionally, unlike LDA-SP Bergsma et al.'s system doesn't produce human-interpretable topics. Finally, we note that LDA-SP and Bergsma's system are potentially complimentary – the output of LDA-SP could be used to generate higher-quality training data for Bergsma, potentially improving their results.

Topic models such as LDA (Blei et al., 2003) and its variants have recently begun to see use in many NLP applications such as summarization (Daumé III and Marcu, 2006), document alignment and segmentation (Chen et al., 2009), and inferring class-attribute hierarchies (Reisinger and Pasca, 2009). Our particular model, LinkLDA, has been applied to a few NLP tasks such as simultaneously modeling the words appearing in blog posts and users who will likely respond to them (Yano et al., 2009), modeling topic-aligned articles in different languages (Mimno et al., 2009), and word sense induction (Brody and Lapata, 2009).

Finally, we highlight two systems, developed independently of our own, which apply LDA-style models to similar tasks. Ó Séaghdha (2010) proposes a series of LDA-style models for the task of computing selectional preferences. This work learns selectional preferences between the following grammatical relations: verb-object, noun-noun, and adjective-noun. It also focuses on jointly modeling the generation of both predicate and argument, and evaluation is performed on a set of human-plausibility judgments obtaining impressive results against Keller and Lapata's (2003) Web hit-count based system. Van Durme and Gildea (2009) proposed applying LDA to general knowledge templates extracted using the KNEXT system (Schubert and Tong, 2003). In contrast, our work uses LinkLDA and focuses on modeling multiple arguments of a relation (*e.g.*, the subject and direct object of a verb).

3 Topic Models for Selectional Prefs.

We present a series of topic models for the task of computing selectional preferences. These models vary in the amount of independence they assume between a_1 and a_2. At one extreme is IndependentLDA, a model which assumes that both a_1 and a_2 are generated completely independently. On the other hand, JointLDA, the model at the other extreme (Figure 1) assumes both arguments of a specific extraction are generated based on a single hidden variable z. LinkLDA (Figure 2) lies between these two extremes, and as demonstrated in Section 4, it is the best model for our relation data.

We are given a set R of binary relations and a corpus $\mathcal{D} = \{r(a_1, a_2)\}$ of extracted instances for these relations. [2] Our task is to compute, for each argument a_i of each relation r, a set of usual argument values (noun phrases) that it takes. For example, for the relation *is headquartered in* the first argument set will include companies like *Microsoft, Intel, General Motors* and second argument will favor locations like *New York, California, Seattle*.

3.1 IndependentLDA

We first describe the straightforward application of LDA to modeling our corpus of extracted relations. In this case two separate LDA models are used to model a_1 and a_2 independently.

In the generative model for our data, each relation r has a corresponding multinomial over topics θ_r, drawn from a Dirichlet. For each extraction, a hidden topic z is first picked according to θ_r, and then the observed argument a is chosen according to the multinomial β_z.

Readers familiar with topic modeling terminology can understand our approach as follows: we treat each relation as a document whose contents consist of a bags of words corresponding to all the noun phrases observed as arguments of the relation in our corpus. Formally, LDA generates each argument in the corpus of relations as follows:

for each topic $t = 1 \ldots T$ **do**
  Generate β_t according to symmetric Dirichlet distribution Dir(η).
end for
for each relation $r = 1 \ldots |R|$ **do**
  Generate θ_r according to Dirichlet distribution Dir(α).
  for each tuple $i = 1 \ldots N_r$ **do**
    Generate $z_{r,i}$ from Multinomial(θ_r).
    Generate the argument $a_{r,i}$ from multinomial $\beta_{z_{r,i}}$.
  end for
end for

One weakness of IndependentLDA is that it doesn't jointly model a_1 and a_2 together. Clearly this is undesirable, as information about which topics one of the arguments favors can help inform the topics chosen for the other. For example, class pairs such as (*team*, *game*), (*politician*, *political issue*) form much more plausible selectional preferences than, say, (*team*, *political issue*), (*politician*, *game*).

[2] We focus on binary relations, though the techniques presented in the paper are easily extensible to n-ary relations.

3.2 JointLDA

As a more tightly coupled alternative, we first propose JointLDA, whose graphical model is depicted in Figure 1. The key difference in JointLDA (versus LDA) is that instead of one, it maintains *two* sets of topics (latent distributions over words) denoted by β and γ, one for classes of each argument. A topic id k represents a pair of topics, β_k and γ_k, that co-occur in the arguments of extracted relations. Common examples include *(Person, Location), (Politician, Political issue), etc.* The hidden variable $z = k$ indicates that the noun phrase for the first argument was drawn from the multinomial β_k, and that the second argument was drawn from γ_k. The per-relation distribution θ_r is a multinomial over the topic ids and represents the selectional preferences, both for arg1s and arg2s of a relation r.

Although JointLDA has many desirable properties, it has some drawbacks as well. Most notably, in JointLDA topics correspond to pairs of multinomials (β_k, γ_k); this leads to a situation in which multiple redundant distributions are needed to represent the same underlying semantic class. For example consider the case where we we need to represent the following selectional preferences for our corpus of relations: (*person*, *location*), (*person*, *organization*), and (*person*, *crime*). Because JointLDA requires a separate pair of multinomials for each topic, it is forced to use 3 separate multinomials to represent the class *person*, rather than learning a single distribution representing *person* and choosing 3 different topics for a_2. This results in poor generalization because the data for a single class is divided into multiple topics.

In order to address this problem while maintaining the sharing of influence between a_1 and a_2, we next present LinkLDA, which represents a compromise between IndependentLDA and JointLDA. LinkLDA is more flexible than JointLDA, allowing different topics to be chosen for a_1, and a_2, however still models the generation of topics from the same distribution for a given relation.

3.3 LinkLDA

Figure 2 illustrates the LinkLDA model in the plate notation, which is analogous to the model in (Erosheva et al., 2004). In particular note that each a_i is drawn from a different hidden topic z_i, however the z_i's are drawn from the same distribution θ_r for a given relation r. To facilitate learning related topic pairs between arguments we employ a sparse prior over the per-relation topic distributions. Because a few topics are likely to be assigned most of the probability mass for a given relation it is more likely (although not necessary) that the same topic number k will be drawn for both arguments.

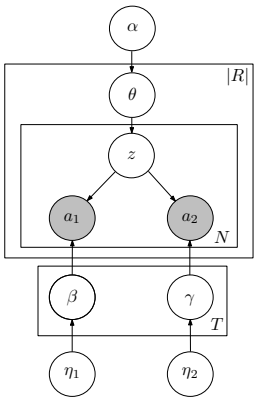


Figure 1: JointLDA

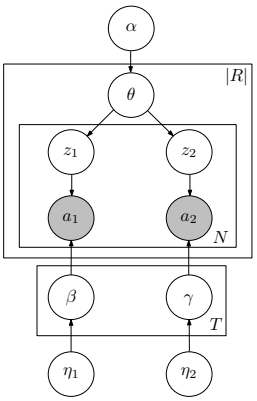


Figure 2: LinkLDA

When comparing LinkLDA with JointLDA the better model may not seem immediately clear. On the one hand, JointLDA jointly models the generation of both arguments in an extracted tuple. This allows one argument to help disambiguate the other in the case of ambiguous relation strings. LinkLDA, however, is more flexible; rather than requiring both arguments to be generated from one of $|Z|$ possible pairs of multinomials (β_z, γ_z), LinkLDA allows the arguments of a given extraction to be generated from $|Z|^2$ possible pairs. Thus, instead of imposing a hard constraint that $z_1 = z_2$ (as in JointLDA), LinkLDA simply assigns a higher probability to states in which $z_1 = z_2$, because both hidden variables are drawn from the same (sparse) distribution θ_r. LinkLDA can thus re-use argument classes, choosing different combinations of topics for the arguments if it fits the data better. In Section 4 we show experimentally that LinkLDA outperforms JointLDA (and IndependentLDA) by wide margins. We use LDA-SP to refer to LinkLDA in all the experiments below.

3.4 Inference

For all the models we use collapsed Gibbs sampling for inference in which each of the hidden variables (*e.g.*, $z_{r,i,1}$ and $z_{r,i,2}$ in LinkLDA) are sampled sequentially conditioned on a full-assignment to all others, integrating out the parameters (Griffiths and Steyvers, 2004). This produces robust parameter estimates, as it allows computation of expectations over the posterior distribution

as opposed to estimating maximum likelihood parameters. In addition, the integration allows the use of sparse priors, which are typically more appropriate for natural language data. In all experiments we use hyperparameters $\alpha = \eta_1 = \eta_2 = 0.1$. We generated initial code for our samplers using the Hierarchical Bayes Compiler (Daume III, 2007).

3.5 Advantages of Topic Models

There are several advantages to using topic models for our task. First, they naturally model the class-based nature of selectional preferences, but don't take a pre-defined set of classes as input. Instead, they compute the classes automatically. This leads to better lexical coverage since the issue of matching a new argument to a known class is side-stepped. Second, the models naturally handle ambiguous arguments, as they are able to assign different topics to the same phrase in different contexts. Inference in these models is also scalable – linear in both the size of the corpus as well as the number of topics. In addition, there are several scalability enhancements such as SparseLDA (Yao et al., 2009), and an approximation of the Gibbs Sampling procedure can be efficiently parallelized (Newman et al., 2009). Finally we note that, once a topic distribution has been learned over a set of training relations, one can efficiently apply inference to unseen relations (Yao et al., 2009).

4 Experiments

We perform three main experiments to assess the quality of the preferences obtained using topic models. The first is a task-independent evaluation using a pseudo-disambiguation experiment (Section 4.2), which is a standard way to evaluate the quality of selectional preferences (Rooth et al., 1999; Erk, 2007; Bergsma et al., 2008). We use this experiment to compare the various topic models as well as the best model with the known state of the art approaches to selectional preferences. Secondly, we show significant improvements to performance at an end-task of textual inference in Section 4.3. Finally, we report on the quality of a large database of Wordnet-based preferences obtained after manually associating our topics with Wordnet classes (Section 4.4).

4.1 Generalization Corpus

For all experiments we make use of a corpus of $r(a_1, a_2)$ tuples, which was automatically extracted by TEXTRUNNER (Banko and Etzioni, 2008) from 500 million Web pages.

To create a *generalization corpus* from this large dataset. We first selected 3,000 relations from the middle of the tail (we used the 2,000-5,000 most frequent ones)[3] and collected all instances. To reduce sparsity, we discarded all tuples containing an NP that occurred fewer than 50 times in the data. This resulted in a vocabulary of about 32,000 noun phrases, and a set of about 2.4 million tuples in our generalization corpus.

We inferred topic-argument and relation-topic multinomials (β, γ, and θ) on the generalization corpus by taking 5 samples at a lag of 50 after a burn in of 750 iterations. Using multiple samples introduces the risk of *topic drift* due to lack of identifiability, however we found this to not be a problem in practice. During development we found that the topics tend to remain stable across multiple samples after sufficient burn in, and multiple samples improved performance. Table 1 lists sample topics and high ranked words for each (for both arguments) as well as relations favoring those topics.

4.2 Task Independent Evaluation

We first compare the three LDA-based approaches to each other and two state of the art similarity based systems (Erk, 2007) (using mutual information and Jaccard similarity respectively). These similarity measures were shown to outperform the generative model of Rooth et al. (1999), as well as class-based methods such as Resnik's. In this pseudo-disambiguation experiment an observed tuple is paired with a pseudo-negative, which has both arguments randomly generated from the whole vocabulary (according to the corpus-wide distribution over arguments). The task is, for each relation-argument pair, to determine whether it is observed, or a random distractor.

4.2.1 Test Set

For this experiment we gathered a primary corpus by first randomly selecting 100 high-frequency relations *not* in the generalization corpus. For each relation we collected all tuples containing arguments in the vocabulary. We held out 500 randomly selected tuples as the test set. For each tu-

[3] Many of the most frequent relations have very weak selectional preferences, and thus provide little signal for inferring meaningful topics. For example, the relations *has* and *is* can take just about any arguments.

Topic t	Arg1	Relations which assign highest probability to t	Arg2
18	The residue - The mixture - The reaction mixture - The solution - the mixture - the reaction mixture - the residue - The reaction - the solution - The filtrate - the reaction - The product - The crude product - The pellet - The organic layer - Thereto - This solution - The resulting solution - Next - The organic phase - The resulting mixture - C.)	was treated with, is treated with, was poured into, was extracted with, was purified by, was diluted with, was filtered through, is disolved in, is washed with	EtOAc - CH2Cl2 - H2O - CH.sub.2Cl.sub.2 - H.sub.2O - water - MeOH - NaHCO3 - Et2O - NHCl - CHCl.sub.3 - NHCl - dropwise - CH2Cl.sub.2 - Celite - Et.sub.2O - Cl.sub.2 - NaOH - AcOEt - CH2C12 - the mixture - saturated NaHCO3 - SiO2 - H2O - N hydrochloric acid - NHCl - preparative HPLC - to0 C
151	the Court - The Court - the Supreme Court - The Supreme Court - this Court - Court - The US Supreme Court - the court - This Court - the US Supreme Court - The court - Supreme Court - Judge - the Court of Appeals - A federal judge	will hear, ruled in, decides, upholds, struck down, overturned, sided with, affirms	the case - the appeal - arguments - a case - evidence - this case - the decision - the law - testimony - the State - an interview - an appeal - cases - the Court - that decision - Congress - a decision - the complaint - oral arguments - a law - the statute
211	President Bush - Bush - The President - Clinton - the President - President Clinton - President George W. Bush - Mr. Bush - The Governor - the Governor - Romney - McCain - The White House - President - Schwarzenegger - Obama	hailed, vetoed, promoted, will deliver, favors, denounced, defended	the bill - a bill - the decision - the war - the idea - the plan - the move - the legislation - legislation - the measure - the proposal - the deal - this bill - a measure - the program - the law - the resolution - efforts - the agreement - gay marriage - the report - abortion
224	Google - Software - the CPU - Clicking - Excel - the user - Firefox - System - The CPU - Internet Explorer - the ability - Program - users - Option - SQL Server - Code - the OS - the BIOS	will display, to store, to load, processes, cannot find, invokes, to search for, to delete	data - files - the data - the file - the URL - information - the files - images - a URL - the information - the IP address - the user - text - the code - a file - the page - IP addresses - PDF files - messages - pages - an IP address

Table 1: Example argument lists from the inferred topics. For each topic number t we list the most probable values according to the multinomial distributions for each argument (β_t and γ_t). The middle column reports a few relations whose inferred topic distributions θ_r assign highest probability to t.

ple $r(a_1, a_2)$ in the held-out set, we removed all tuples in the training set containing either of the *rel-arg* pairs, *i.e.*, any tuple matching $r(a_1, *)$ or $r(*, a_2)$. Next we used collapsed Gibbs sampling to infer a distribution over topics, θ_r, for each of the relations in the primary corpus (based solely on tuples in the training set) using the topics from the generalization corpus.

For each of the 500 observed tuples in the test-set we generated a pseudo-negative tuple by randomly sampling two noun phrases from the distribution of NPs in both corpora.

4.2.2 Prediction

Our prediction system needs to determine whether a specific relation-argument pair is admissible according to the selectional preferences or is a random distractor (D). Following previous work, we perform this experiment independently for the two relation-argument pairs (r, a_1) and (r, a_2).

We first compute the probability of observing a_1 for first argument of relation r given that it is not a distractor, $P(a_1|r, \neg D)$, which we approximate by its probability given an estimate of the parameters inferred by our model, marginalizing over hidden topics t. The analysis for the second argument is similar.

$$\begin{aligned} P(a_1|r, \neg D) \approx P_{LDA}(a_1|r) &= \sum_{t=0}^{T} P(a_1|t)P(t|r) \\ &= \sum_{t=0}^{T} \beta_t(a_1)\theta_r(t) \end{aligned}$$

A simple application of Bayes Rule gives the probability that a particular argument is not a distractor. Here the distractor-related probabilities are independent of r, *i.e.*, $P(D|r) = P(D)$, $P(a_1|D, r) = P(a_1|D)$, *etc.* We estimate $P(a_1|D)$ according to their frequency in the generalization corpus.

$$\begin{aligned} P(\neg D|r, a_1) &= \frac{P(\neg D|r)P(a_1|r, \neg D)}{P(a_1|r)} \\ &\approx \frac{P(\neg D)P_{LDA}(a_1|r)}{P(D)P(a_1|D) + P(\neg D)P_{LDA}(a_1|r)} \end{aligned}$$

4.2.3 Results

Figure 3 plots the precision-recall curve for the pseudo-disambiguation experiment comparing the three different topic models. LDA-SP, which uses LinkLDA, substantially outperforms both IndependentLDA and JointLDA.

Next, in figure 4, we compare LDA-SP with mutual information and Jaccard similarities using both the generalization and primary corpus for

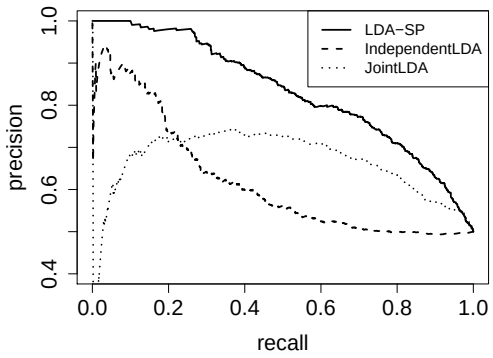


Figure 3: Comparison of LDA-based approaches on the pseudo-disambiguation task. LDA-SP (LinkLDA) substantially outperforms the other models.

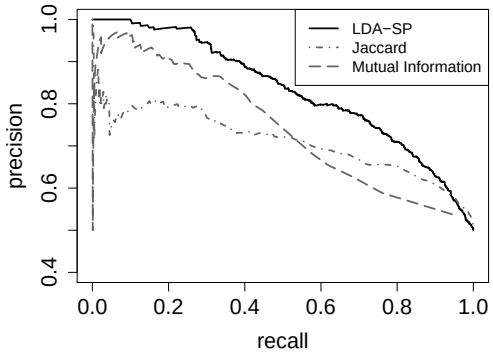


Figure 4: Comparison to similarity-based selectional preference systems. LDA-SP obtains 85% higher recall at precision 0.9.

computation of similarities. We find LDA-SP significantly outperforms these methods. Its edge is most noticed at high precisions; it obtains 85% more recall at 0.9 precision compared to mutual information. Overall LDA-SP obtains an 15% increase in the area under precision-recall curve over mutual information. All three systems' AUCs are shown in Table 2; LDA-SP's improvements over both Jaccard and mutual information are highly significant with a significance level less than 0.01 using a paired t-test.

In addition to a superior performance in selectional preference evaluation LDA-SP also produces a set of coherent topics, which can be useful in their own right. For instance, one could use them for tasks such as set-expansion (Carlson et al., 2010) or automatic thesaurus induction (Etzioni et al., 2005; Kozareva et al., 2008).

	LDA-SP	MI-Sim	Jaccard-Sim
AUC	0.833	0.727	0.711

Table 2: Area under the precision recall curve. LDA-SP's AUC is significantly higher than both similarity-based methods according to a paired t-test with a significance level below 0.01.

4.3 End Task Evaluation

We now evaluate LDA-SP's ability to improve performance at an end-task. We choose the task of improving textual entailment by learning selectional preferences for inference rules and filtering inferences that do not respect these. This application of selectional preferences was introduced by Pantel *et. al.* (2007). For now we stick to inference rules of the form $r_1(a_1, a_2) \Rightarrow r_2(a_1, a_2)$, though our ideas are more generally applicable to more complex rules. As an example, the rule (X *defeats* Y) $\Rightarrow$ (X *plays* Y) holds when X and Y are both sports teams, however fails to produce a reasonable inference if X and Y are *Britain* and *Nazi Germany* respectively.

4.3.1 Filtering Inferences

In order for an inference to be plausible, both relations must have similar selectional preferences, and further, the arguments must obey the selectional preferences of both the antecedent r_1 and the consequent r_2.[4] Pantel et al. (2007) made use of these intuitions by producing a set of class-based selectional preferences for each relation, then filtering out any inferences where the arguments were incompatible with the intersection of these preferences. In contrast, we take a probabilistic approach, evaluating the quality of a specific inference by measuring the probability that the arguments in both the antecedent and the consequent were drawn from the same hidden topic in our model. Note that this probability captures both the requirement that the antecedent and consequent have similar selectional preferences, and that the arguments from a particular instance of the rule's application match their overlap.

We use $z_{r_i,j}$ to denote the topic that generates the j^{th} argument of relation r_i. The probability that the two arguments a_1, a_2 were drawn from the same hidden topic factorizes as follows due to the conditional independences in our model:[5]

$$P(z_{r_1,1} = z_{r_2,1}, z_{r_1,2} = z_{r_2,2}|a_1, a_2) = P(z_{r_1,1} = z_{r_2,1}|a_1)P(z_{r_1,2} = z_{r_2,2}|a_2)$$

[4] Similarity-based and discriminative methods are not applicable to this task as they offer no straightforward way to compare the similarity between selectional preferences of two relations.

[5] Note that all probabilities are conditioned on an estimate of the parameters θ, β, γ from our model, which are omitted for compactness.

To compute each of these factors we simply marginalize over the hidden topics:

$$P(z_{r_1,j} = z_{r_2,j}|a_j) = \sum_{t=1}^{T} P(z_{r_1,j} = t|a_j)P(z_{r_2,j} = t|a_j)$$

where $P(z = t|a)$ can be computed using Bayes rule. For example,

$$\begin{aligned} P(z_{r_1,1} = t|a_1) &= \frac{P(a_1|z_{r_1,1} = t)P(z_{r_1,1} = t)}{P(a_1)} \\ &= \frac{\beta_t(a_1)\theta_{r_1}(t)}{P(a_1)} \end{aligned}$$

4.3.2 Experimental Conditions

In order to evaluate LDA-SP's ability to filter inferences based on selectional preferences we need a set of inference rules between the relations in our corpus. We therefore mapped the DIRT Inference rules (Lin and Pantel, 2001), (which consist of pairs of dependency paths) to TEXTRUNNER relations as follows. We first gathered all instances in the generalization corpus, and for each $r(a_1, a_2)$ created a corresponding simple sentence by concatenating the arguments with the relation string between them. Each such simple sentence was parsed using Minipar (Lin, 1998). From the parses we extracted all dependency paths between nouns that contain only words present in the TEXTRUNNER relation string. These dependency paths were then matched against each pair in the DIRT database, and all pairs of associated relations were collected producing about 26,000 inference rules.

Following Pantel et al. (2007) we randomly sampled 100 inference rules. We then automatically filtered out any rules which contained a negation, or for which the antecedent and consequent contained a pair of antonyms found in WordNet (this left us with 85 rules). For each rule we collected 10 random instances of the antecedent, and generated the consequent. We randomly sampled 300 of these inferences to hand-label.

4.3.3 Results

In figure 5 we compare the precision and recall of LDA-SP against the top two performing systems described by Pantel et al. (ISP.IIM-$\vee$ and ISP.JIM, both using the CBC clusters (Pantel, 2003)). We find that LDA-SP achieves both higher precision and recall than ISP.IIM-$\vee$. It is also able to achieve the high-precision point of ISP.JIM and can trade precision to get a much larger recall.

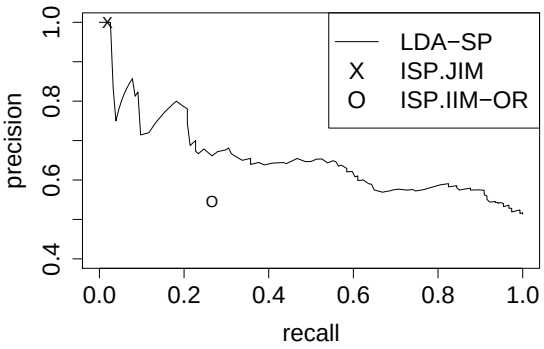


Figure 5: Precision and recall on the inference filtering task.

Top 10 Inference Rules Ranked by LDA-SP		
antecedent	consequent	KL-div
will begin at	will start at	0.014999
shall review	shall determine	0.129434
may increase	may reduce	0.214841
walk from	walk to	0.219471
consume	absorb	0.240730
shall keep	shall maintain	0.264299
shall pay to	will notify	0.290555
may apply for	may obtain	0.313916
copy	download	0.316502
should pay	must pay	0.371544
Bottom 10 Inference Rules Ranked by LDA-SP		
antecedent	consequent	KL-div
lose to	shall take	10.011848
should play	could do	10.028904
could play	get in	10.048857
will start at	move to	10.060994
shall keep	will spend	10.105493
should play	get in	10.131299
shall pay to	leave for	10.131364
shall keep	return to	10.149797
shall keep	could do	10.178032
shall maintain	have spent	10.221618

Table 3: Top 10 and Bottom 10 ranked inference rules ranked by LDA-SPafter automatically filtering out negations and antonyms (using WordNet).

In addition we demonstrate LDA-SP's ability to rank inference rules by measuring the Kullback Leibler Divergence[6] between the topic-distributions of the antecedent and consequent, θ_{r_1} and θ_{r_2} respectively. Table 3 shows the top 10 and bottom 10 rules out of the 26,000 ranked by KL Divergence after automatically filtering antonyms (using WordNet) and negations. For slight variations in rules (*e.g.*, symmetric pairs) we mention only one example to show more variety.

[6]KL-Divergence is an information-theoretic measure of the similarity between two probability distributions, and defined as follows: $KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$.

4.4 A Repository of Class-Based Preferences

Finally we explore LDA-SP's ability to produce a repository of human interpretable class-based selectional preferences. As an example, for the relation *was born in*, we would like to infer that the plausible arguments include (*person*, *location*) and (*person*, *date*).

Since we already have a set of topics, our task reduces to mapping the inferred topics to an equivalent class in a taxonomy (*e.g.*, WordNet). We experimented with automatic methods such as Resnik's, but found them to have all the same problems as directly applying these approaches to the SP task.[7] Guided by the fact that we have a relatively small number of topics (600 total, 300 for each argument) we simply chose to label them manually. By labeling this small number of topics we can infer class-based preferences for an arbitrary number of relations.

In particular, we applied a semi-automatic scheme to map topics to WordNet. We first applied Resnik's approach to automatically shortlist a few candidate WordNet classes for each topic. We then manually picked the best class from the shortlist that best represented the 20 top arguments for a topic (similar to Table 1). We marked all incoherent topics with a special symbol $\emptyset$. This process took one of the authors about 4 hours to complete.

To evaluate how well our topic-class associations carry over to unseen relations we used the same random sample of 100 relations from the pseudo-disambiguation experiment.[8] For each argument of each relation we picked the top two topics according to frequency in the 5 Gibbs samples. We then discarded any topics which were labeled with $\emptyset$; this resulted in a set of 236 predictions. A few examples are displayed in table 4.

We evaluated these classes and found the accuracy to be around 0.88. We contrast this with Pantel's repository,[9] the only other released database of selectional preferences to our knowledge. We evaluated the same 100 relations from his website and tagged the top 2 classes for each argument and evaluated the accuracy to be roughly 0.55.

[7] Perhaps recent work on automatic coherence ranking (Newman et al., 2010) and labeling (Mei et al., 2007) could produce better results.

[8] Recall that these 100 were not part of the original 3,000 in the generalization corpus, and are, therefore, representative of new "unseen" relations.

[9] http://demo.patrickpantel.com/Content/LexSem/paraphrase.htm

arg1 class	relation	arg2 class
politician#1	was running for	leader#1
people#1	will love	show#3
organization#1	has responded to	accusation#2
administrative_unit#1	has appointed	administrator#3

Table 4: Class-based Selectional Preferences.

We emphasize that tagging a pair of class-based preferences is a highly subjective task, so these results should be treated as preliminary. Still, these early results are promising. We wish to undertake a larger scale study soon.

5 Conclusions and Future Work

We have presented an application of topic modeling to the problem of automatically computing selectional preferences. Our method, LDA-SP, learns a distribution over topics for each relation while simultaneously grouping related words into these topics. This approach is capable of producing human interpretable classes, however, avoids the drawbacks of traditional class-based approaches (poor lexical coverage and ambiguity). LDA-SP achieves state-of-the-art performance on predictive tasks such as pseudo-disambiguation, and filtering incorrect inferences.

Because LDA-SP generates a complete probabilistic model for our relation data, its results are easily applicable to many other tasks such as identifying similar relations, ranking inference rules, *etc.* In the future, we wish to apply our model to automatically discover new inference rules and paraphrases.

Finally, our repository of selectional preferences for 10,000 relations is available at http://www.cs.washington.edu/research/ldasp.

Acknowledgments

We would like to thank Tim Baldwin, Colin Cherry, Jesse Davis, Elena Erosheva, Stephen Soderland, Dan Weld, in addition to the anonymous reviewers for helpful comments on a previous draft. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, DARPA contract FA8750-09-C-0179, a National Defense Science and Engineering Graduate (NDSEG) Fellowship 32 CFR 168a, and carried out at the University of Washington's Turing Center.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL-08: HLT*.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *EMNLP*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *EACL*, pages 103–111, Morristown, NJ, USA. Association for Computational Linguistics.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *WSDM 2010*.

Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. 2009. Global models of document structure using latent permutations. In *NAACL*.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Comput. Linguist.*

Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. In *Machine Learning*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Hal Daume III. 2007. hbc: Hierarchical bayes compiler. http://hal3.name/hbc.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Elena Erosheva, Stephen Fienberg, and John Lafferty. 2004. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences of the United States of America*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana maria Popescu, Tal Shaked, Stephen Soderl, Daniel S. Weld, and Alex Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.*

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proc Natl Acad Sci U S A*.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Comput. Linguist.*

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL-08: HLT*.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Comput. Linguist.*

Dekang Lin and Patrick Pantel. 2001. Dirt-discovery of inference rules from text. In *KDD*.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proc. Workshop on the Evaluation of Parsing Systems*.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *KDD*.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *EMNLP*.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *JMLR*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *NAACL-HLT*.

Diarmuid Ó Séaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard H. Hovy. 2007. Isp: Learning inferential selectional preferences. In *HLT-NAACL*.

Patrick Andre Pantel. 2003. *Clustering by committee*. Ph.D. thesis, University of Alberta, Edmonton, Alta., Canada.

Joseph Reisinger and Marius Pasca. 2009. Latent variable models of concept-attribute attachment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

P. Resnik. 1996. Selectional constraints: an information-theoretic model and its computational realization. *Cognition*.

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

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*.

Lenhart Schubert and Matthew Tong. 2003. Extracting and evaluating general world knowledge from the brown corpus. In *In Proc. of the HLT-NAACL Workshop on Text Meaning*, pages 7–13.

Benjamin Van Durme and Daniel Gildea. 2009. Topic models for corpus-centric knowledge generalization. In *Technical Report TR-946, Department of Computer Science, University of Rochester, Rochester*.

Tae Yano, William W. Cohen, and Noah A. Smith. 2009. Predicting response to political blog posts with topic models. In *NAACL*.

L. Yao, D. Mimno, and A. Mccallum. 2009. Efficient methods for topic model inference on streaming document collections. In *KDD*.

Latent variable models of selectional preference

Diarmuid Ó Séaghdha
University of Cambridge
Computer Laboratory
United Kingdom
do242@cl.cam.ac.uk

Abstract

This paper describes the application of so-called *topic models* to selectional preference induction. Three models related to Latent Dirichlet Allocation, a proven method for modelling document-word co-occurrences, are presented and evaluated on datasets of human plausibility judgements. Compared to previously proposed techniques, these models perform very competitively, especially for infrequent predicate-argument combinations where they exceed the quality of Web-scale predictions while using relatively little data.

1 Introduction

Language researchers have long been aware that many words place semantic restrictions on the words with which they can co-occur in a syntactic relationship. Violations of these restrictions make the sense of a sentence odd or implausible:

(1) Colourless green ideas sleep furiously.

(2) The deer shot the hunter.

Recognising whether or not a selectional restriction is satisfied can be an important trigger for metaphorical interpretations (Wilks, 1978) and also plays a role in the time course of human sentence processing (Rayner et al., 2004). A more relaxed notion of *selectional preference* captures the idea that certain classes of entities are more likely than others to fill a given argument slot of a predicate. In Natural Language Processing, knowledge about probable, less probable and wholly infelicitous predicate-argument pairs is of value for numerous applications, for example semantic role labelling (Gildea and Jurafsky, 2002; Zapirain et al., 2009). The notion of selectional preference is not restricted to surface-level predicates such as verbs and modifiers, but also extends to semantic frames (Erk, 2007) and inference rules (Pantel et al., 2007).

The fundamental problem that selectional preference models must address is data sparsity: in many cases insufficient corpus data is available to reliably measure the plausibility of a predicate-argument pair by counting its observed frequency. A rarely seen pair may be fundamentally implausible (*a carrot laughed*) or plausible but rarely expressed (*a manservant laughed*).[1] In general, it is beneficial to smooth plausibility estimates by integrating knowledge about the frequency of other, similar predicate-argument pairs. The task thus share some of the nature of language modelling; however, it is a task less amenable to approaches that require very large training corpora and one where the semantic quality of a model is of greater importance.

This paper takes up tools ("topic models") that have been proven successful in modelling document-word co-occurrences and adapts them to the task of selectional preference learning. Advantages of these models include a well-defined generative model that handles sparse data well, the ability to jointly induce semantic classes and predicate-specific distributions over those classes, and the enhanced statistical strength achieved by sharing knowledge across predicates. Section 2 surveys prior work on selectional preference modelling and on semantic applications of topic models. Section 3 describes the models used in our experiments. Section 4 provides details of the experimental design. Section 5 presents results for our models on the task of predicting human plausibility judgements for predicate-argument combinations; we show that performance is generally competi-

[1] At time of writing, Google estimates 855 hits for "*a|the carrot|carrots laugh|laughs|laughed*" and 0 hits for "*a|the manservant|manservants|menservants laugh|laughs|laughed*"; many of the *carrot* hits are false positives but a significant number are true subject-verb observations.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 435–444,
Uppsala, Sweden, 11-16 July 2010.

tive with or superior to a number of other models, including models using Web-scale resources, especially for low-frequency examples. In Section 6 we wrap up by summarising the paper's conclusions and sketching directions for future research.

2 Related work

2.1 Selectional preference learning

The representation (and latterly, learning) of selectional preferences for verbs and other predicates has long been considered a fundamental problem in computational semantics (Resnik, 1993). Many approaches to the problem use lexical taxonomies such as WordNet to identify the semantic classes that typically fill a particular argument slot for a predicate (Resnik, 1993; Clark and Weir, 2002; Schulte im Walde et al., 2008). In this paper, however, we focus on methods that do not assume the availability of a comprehensive taxonomy but rather induce semantic classes automatically from a corpus of text. Such methods are more generally applicable, for example in domains or languages where handbuilt semantic lexicons have insufficient coverage or are non-existent.

Rooth et al. (1999) introduced a model of selectional preference induction that casts the problem in a probabilistic latent-variable framework. In Rooth et al.'s model each observed predicate-argument pair is probabilistically generated from a latent variable, which is itself generated from an underlying distribution on variables. The use of latent variables, which correspond to coherent clusters of predicate-argument interactions, allow probabilities to be assigned to predicate-argument pairs which have not previously been observed by the model. The discovery of these predicate-argument clusters and the estimation of distributions on latent and observed variables are performed simultaneously via an Expectation Maximisation procedure. The work presented in this paper is inspired by Rooth et al.'s latent variable approach, most directly in the model described in Section 3.3. Erk (2007) and Padó et al. (2007) describe a corpus-driven smoothing model which is not probabilistic in nature but relies on similarity estimates from a "semantic space" model that identifies semantic similarity with closeness in a vector space of co-occurrences. Bergsma et al. (2008) suggest learning selectional preferences in a discriminative way, by training a collection of SVM classifiers to recognise likely and unlikely arguments for predicates of interest.

Keller and Lapata (2003) suggest a simple alternative to smoothing-based approaches. They demonstrate that noisy counts from a Web search engine can yield estimates of plausibility for predicate-argument pairs that are superior to models learned from a smaller parsed corpus. The assumption inherent in this approach is that given sufficient text, all plausible predicate-argument pairs will be observed with frequency roughly correlated with their degree of plausibility. While the model is undeniably straightforward and powerful, it has a number of drawbacks: it presupposes an extremely large corpus, the like of which will only be available for a small number of domains and languages, and it is only suitable for relations that are identifiable by searching raw text for specific lexical patterns.

2.2 Topic modelling

The task of inducing coherent semantic clusters is common to many research areas. In the field of document modelling, a class of methods known as "topic models" have become a de facto standard for identifying semantic structure in documents. These include the Latent Dirichlet Allocation (LDA) model of Blei et al. (2003) and the Hierarchical Dirichlet Process model of Teh et al. (2006). Formally seen, these are hierarchical Bayesian models which induce a set of latent variables or topics that are shared across documents. The combination of a well-defined probabilistic model and Gibbs sampling procedure for estimation guarantee (eventual) convergence and the avoidance of degenerate solutions. As a result of intensive research in recent years, the behaviour of topic models is well-understood and computationally efficient implementations have been developed. The tools provided by this research are used in this paper as the building blocks of our selectional preference models.

Hierarchical Bayesian modelling has recently gained notable popularity in many core areas of natural language processing, from morphological segmentation (Goldwater et al., 2009) to opinion modelling (Lin et al., 2006). Yet so far there have been relatively few applications to traditional lexical semantic tasks. Boyd-Graber et al. (2007) integrate a model of random walks on the WordNet graph into an LDA topic model to build an unsupervised word sense disambiguation system. Brody

and Lapata (2009) adapt the basic LDA model for application to unsupervised word sense induction; in this context, the topics learned by the model are assumed to correspond to distinct senses of a particular lemma. Zhang et al. (2009) are also concerned with inducing multiple senses for a particular term; here the goal is to identify distinct entity types in the output of a pattern-based entity set discovery system. Reisinger and Paşca (2009) use LDA-like models to map automatically acquired attribute sets onto the WordNet hierarchy. Griffiths et al. (2007) demonstrate that topic models learned from document-word co-occurrences are good predictors of semantic association judgements by humans.

Simultaneously to this work, Ritter et al. (2010) have also investigated the use of topic models for selectional preference learning. Their goal is slightly different to ours in that they wish to model the probability of a binary predicate taking two specified arguments, i.e., $P(n_1, n_2|v)$, whereas we model the joint and conditional probabilities of a predicate taking a single specified argument. The model architecture they propose, LinkLDA, falls somewhere between our LDA and DUAL-LDA models. Hence LinkLDA could be adapted to estimate $P(n, v|r)$ as DUAL-LDA does, but a preliminary investigation indicates that it does not perform well in this context. The most likely explanation is that LinkLDA generates its two arguments independently, which may be suitable for distinct argument positions of a given predicate but is unsuitable when one of those "arguments" is in fact the predicate.

The models developed in this paper, though intended for semantic modelling, also bear some similarity to the internals of generative syntax models such as the "infinite tree" (Finkel et al., 2007). In some ways, our models are less ambitious than comparable syntactic models as they focus on specific fragments of grammatical structure rather than learning a more general representation of sentence syntax. It would be interesting to evaluate whether this restricted focus improves the quality of the learned model or whether general syntax models can also capture fine-grained knowledge about combinatorial semantics.

3 Three selectional preference models

3.1 Notation

In the model descriptions below we assume a predicate vocabulary of V types, an argument vocabulary of N types and a relation vocabulary of R types. Each predicate type is associated with a singe relation; for example the predicate type *eat:V:dobj* (the direct object of the verb *eat*) is treated as distinct from *eat:V:subj* (the subject of the verb *eat*). The training corpus consists of W observations of argument-predicate pairs. Each model has at least one vocabulary of Z arbitrarily labelled latent variables. f_{zn} is the number of observations where the latent variable z has been associated with the argument type n, f_{zv} is the number of observations where z has been associated with the predicate type v and f_{zr} is the number of observations where z has been associated with the relation r. Finally, $f_{z\cdot}$ is the total number of observations associated with z and $f_{\cdot v}$ is the total number of observations containing the predicate v.

3.2 Latent Dirichlet Allocation

As noted above, LDA was originally introduced to model sets of documents in terms of topics, or clusters of terms, that they share in varying proportions. For example, a research paper on bioinformatics may use some vocabulary that is shared with general computer science papers and some vocabulary that is shared with biomedical papers. The analogical move from modelling document-term cooccurrences to modelling predicate-argument cooccurrences is intuitive: we assume that each predicate is associated with a distribution over semantic classes ("topics") and that these classes are shared across predicates. The high-level "generative story" for the LDA selectional preference model is as follows:

(1) For each predicate v, draw a multinomial distribution Θ_v over argument classes from a Dirichlet distribution with parameters α.

(2) For each argument class z, draw a multinomial distribution Φ_z over argument types from a Dirichlet with parameters β.

(3) To generate an argument for v, draw an argument class z from Θ_v and then draw an argument type n from Φ_z

The resulting model can be written as:

$$P(n|v, r) = \sum_z P(n|z)P(z|v, r) \quad (1)$$

$$\propto \sum_z \frac{f_{zn} + \beta}{f_{z\cdot} + N\beta} \frac{f_{zv} + \alpha_z}{f_{\cdot v} + \sum_{z'} \alpha_{z'}} \quad (2)$$

Due to multinomial-Dirichlet conjugacy, the distributions Θ_v and Φ_z can be integrated out and do not appear explicitly in the above formula. The first term in (2) can be seen as a smoothed estimate of the probability that class z produces the argument n; the second is a smoothed estimate of the probability that predicate v takes an argument belonging to class z. One important point is that the smoothing effects of the Dirichlet priors on Θ_v and Φ_z are greatest for predicates and arguments that are rarely seen, reflecting an intuitive lack of certainty. We assume an asymmetric Dirichlet prior on Θ_v (the α parameters can differ for each class) and a symmetric prior on Φ_z (all β parameters are equal); this follows the recommendations of Wallach et al. (2009) for LDA. This model estimates predicate-argument probabilities conditional on a given predicate v; it cannot by itself provide joint probabilities $P(n, v|r)$, which are needed for our plausibility evaluation.

Given a dataset of predicate-argument combinations and values for the hyperparameters α and β, the probability model is determined by the class assignment counts f_{zn} and f_{zv}. Following Griffiths and Steyvers (2004), we estimate the model by Gibbs sampling. This involves resampling the topic assignment for each observation in turn using probabilities estimated from all other observations. One efficiency bottleneck in the basic sampler described by Griffiths and Steyvers is that the entire set of topics must be iterated over for each observation. Yao et al. (2009) propose a reformulation that removes this bottleneck by separating the probability mass $p(z|n, v)$ into a number of buckets, some of which only require iterating over the topics currently assigned to instances of type n, typically far fewer than the total number of topics. It is possible to apply similar reformulations to the models presented in Sections 3.3 and 3.4 below; depending on the model and parameterisation this can reduce the running time dramatically.

Unlike some topic models such as HDP (Teh et al., 2006), LDA is *parametric*: the number of topics Z must be set by the user in advance. However, Wallach et al. (2009) demonstrate that LDA is relatively insensitive to larger-than-necessary choices of Z when the Dirichlet parameters α are optimised as part of model estimation. In our implementation we use the optimisation routines provided as part of the Mallet library, which use an iterative procedure to compute a maximum likelihood estimate of these hyperparameters.[2]

3.3 A Rooth et al.-inspired model

In Rooth et al.'s (1999) selectional preference model, a latent variable is responsible for generating both the predicate and argument types of an observation. The basic LDA model can be extended to capture this kind of predicate-argument interaction; the generative story for the resulting ROOTH-LDA model is as follows:

(1) For each relation r, draw a multinomial distribution Θ_r over interaction classes from a Dirichlet distribution with parameters α.

(2) For each class z, draw a multinomial Φ_z over argument types from a Dirichlet distribution with parameters β and a multinomial Ψ_z over predicate types from a Dirichlet distribution with parameters γ.

(3) To generate an observation for r, draw a class z from Θ_r, then draw an argument type n from Φ_z and a predicate type v from Ψ_z.

The resulting model can be written as:

$$P(n, v|r) = \sum_z P(n|z)P(v|z)P(z|r) \quad (3)$$

$$\propto \sum_z \frac{f_{zn} + \beta}{f_{z\cdot} + N\beta} \frac{f_{zv} + \gamma}{f_{z\cdot} + V\gamma} \frac{f_{zr} + \alpha_z}{f_{\cdot r} + \sum_{z'} \alpha_{z'}} \quad (4)$$

As suggested by the similarity between (4) and (2), the ROOTH-LDA model can be estimated by an LDA-like Gibbs sampling procedure.

Unlike LDA, ROOTH-LDA does model the joint probability $P(n, v|r)$ of a predicate and argument co-occurring. Further differences are that information about predicate-argument co-occurrence is only shared within a given interaction class rather than across the whole dataset and that the distribution Φ_z is not specific to the predicate v but rather to the relation r. This could potentially lead to a loss of model quality, but in practice the ability to induce "tighter" clusters seems to counteract any deterioration this causes.

3.4 A "dual-topic" model

In our third model, we attempt to combine the advantages of LDA and ROOTH-LDA by clustering arguments and predicates according to separate

[2] http://mallet.cs.umass.edu/

class vocabularies. Each observation is generated by two latent variables rather than one, which potentially allows the model to learn more flexible interactions between arguments and predicates.:

(1) For each relation r, draw a multinomial distribution Ξ_r over predicate classes from a Dirichlet with parameters κ.

(2) For each predicate class c, draw a multinomial Ψ_c over predicate types and a multinomial Θ_c over argument classes from Dirichlets with parameters γ and α respectively.

(3) For each argument class z, draw a multinomial distribution Φ_z over argument types from a Dirichlet with parameters β.

(4) To generate an observation for r, draw a predicate class c from Ξ_r, a predicate type from Ψ_c, an argument class z from Θ_c and an argument type from Φ_z.

The resulting model can be written as:

$$P(n,v|r) = \sum_c \sum_z P(n|z)P(z|c)P(v|c)P(c|r) \quad (5)$$

$$\propto \sum_c \sum_z \frac{f_{zn}+\beta}{f_{z\cdot}+N\beta} \frac{f_{zc}+\alpha_z}{f_{\cdot c}+\sum_{z'}\alpha_{z'}} \times \frac{f_{cv}+\gamma}{f_{c\cdot}+V\gamma} \frac{f_{cr}+\kappa_c}{f_{\cdot r}+\sum_{c'}\kappa_{c'}} \quad (6)$$

To estimate this model, we first resample the class assignments for all arguments in the data and then resample class assignments for all predicates. Other approaches are possible – resampling argument and then predicate class assignments for each observation in turn, or sampling argument and predicate assignments together by blocked sampling – though from our experiments it does not seem that the choice of scheme makes a significant difference.

4 Experimental setup

In the document modelling literature, probabilistic topic models are often evaluated on the likelihood they assign to unseen documents; however, it has been shown that higher log likelihood scores do not necessarily correlate with more semantically coherent induced topics (Chang et al., 2009). One popular method for evaluating selectional preference models is by testing the correlation between their predictions and human judgements of plausibility on a dataset of predicate-argument pairs. This can be viewed as a more semantically relevant measurement of model quality than likelihood-based methods, and also permits comparison with non-probabilistic models. In Section 5, we use two plausibility datasets to evaluate our models and compare to other previously published results.

We trained our models on the 90-million word written component of the British National Corpus (Burnard, 1995), parsed with the RASP toolkit (Briscoe et al., 2006). Predicates occurring with just one argument type were removed, as were all tokens containing non-alphabetic characters; no other filtering was done. The resulting datasets consisted of 3,587,172 verb-object observations with 7,954 predicate types and 80,107 argument types, 3,732,470 noun-noun observations with 68,303 predicate types and 105,425 argument types, and 3,843,346 adjective-noun observations with 29,975 predicate types and 62,595 argument types.

During development we used the verb-noun plausibility dataset from Padó et al. (2007) to direct the design of the system. Unless stated otherwise, all results are based on runs of 1,000 iterations with 100 classes, with a 200-iteration burnin period after which hyperparameters were reestimated every 50 iterations.[3] The probabilities estimated by the models ($P(n|v,r)$ for LDA and $P(n,v|r)$ for ROOTH- and DUAL-LDA) were sampled every 50 iterations post-burnin and averaged over three runs to smooth out variance. To compare plausibility scores for different predicates, we require the joint probability $P(n,v|r)$; as LDA does not provide this, we approximate $P_{LDA}(n,v|r) = P_{BNC}(v|r)P_{LDA}(n|v,r)$, where $P_{BNC}(v|r)$ is proportional to the frequency with which predicate v is observed as an instance of relation r in the BNC.

For comparison, we reimplemented the methods of Rooth et al. (1999) and Padó et al. (2007). As mentioned above, Rooth et al. use a latent-variable model similar to (4) but without priors, trained via EM. Our implementation (henceforth ROOTH-EM) chooses the number of classes from the range $(20, 25, \ldots, 50)$ through 5-fold cross-validation on a held-out log-likelihood measure. Settings outside this range did not give good results. Again, we run for 1,000 iterations and average predictions over

[3] These settings were based on the MALLET defaults; we have not yet investigated whether modifying the simulation length or burnin period is beneficial.

LDA	0	Nouns:	*agreement*, *contract*, *permission*, *treaty*, *deal*, ...
	1	Nouns	*information*, *datum*, *detail*, *evidence*, *material*, ...
	2	Nouns	*skill, knowledge, country, technique, understanding*, ...
ROOTH-LDA	0	Nouns	*force, team, army, group, troops*, ...
	0	Verbs	*join, arm, lead, beat, send*, ...
	1	Nouns	*door, eye, mouth, window, gate*, ...
	1	Verbs	*open, close, shut, lock, slam*, ...
DUAL-LDA	0N	Nouns	*house, building, site, home, station*, ...
	1N	Nouns	*stone, foot, bit, breath, line*, ...
	0V	Verbs	*involve, join, lead, represent, concern*, ...
	1V	Verbs	*see, break, have, turn, round*, ...
ROOTH-EM	0	Nouns	*system, method, technique, skill, model*, ...
	0	Verbs	*use, develop, apply, design, introduce*, ...
	1	Nouns	*eye, door, page, face, chapter*,...
	1	Verbs	*see, open, close, watch, keep*,...

Table 1: Most probable words for sample semantic classes induced from verb-object observations

three runs. Padó et al. (2007), a refinement of Erk (2007), is a non-probabilistic method that smooths predicate-argument counts with counts for other observed arguments of the same predicate, weighted by the similarity between arguments. Following their description, we use a 2,000-dimensional space of syntactic co-occurrence features appropriate to the relation being predicted, weight features with the G^2 transformation and compute similarity with the cosine measure.

5 Results

5.1 Induced semantic classes

Table 1 shows sample semantic classes induced by models trained on the corpus of BNC verb-object co-occurrences. LDA clusters nouns only, while ROOTH-LDA and ROOTH-EM learn classes that generate both nouns and verbs and DUAL-LDA clusters nouns and verbs separately. The LDA clusters are generally sensible: class 0 is exemplified by *agreement* and *contract* and class 1 by *information* and *datum*. There are some unintuitive blips, for example *country* appears between *knowledge* and *understanding* in class 2. The ROOTH-LDA classes also feel right: class 0 deals with nouns such as *force*, *team* and *army* which one might *join*, *arm* or *lead* and class 1 corresponds to "things that can be opened or closed" such as a *door*, an *eye* or a *mouth* (though the model also makes the questionable prediction that all these items can plausibly be locked or slammed). The DUAL-LDA classes are notably less coherent, especially when it comes to clustering verbs: DUAL-LDA's class 0V, like ROOTH-LDA's class 0, has verbs that take groups as objects but its class 1V mixes sensible conflations (*turn*, *round*) with very common verbs such as *see* and *have* and the unrelated *break*. The general impression given by inspection of the DUAL-LDA model is that it has problems with mixing and does not manage to learn a good model; we have tried a number of solutions (e.g., blocked sampling of argument and predicate classes), without overcoming this brittleness. Unsurprisingly, ROOTH-EM's classes have a similar feel to ROOTH-LDA; our general impression is that some of ROOTH-EM's classes look even more coherent than the LDA-based models, presumably because it does not use priors to smooth its per-class distributions.

5.2 Comparison with Keller and Lapata (2003)

Keller and Lapata (2003) collected a dataset of human plausibility judgements for three classes of grammatical relation: verb-object, noun-noun modification and adjective-noun modification. The items in this dataset were not chosen to balance plausibility and implausibility (as in prior psycholinguistic experiments) but according to their corpus frequency, leading to a more realistic task. 30 predicates were selected for each relation; each predicate was matched with three arguments from different co-occurrence bands in the BNC, e.g., *naughty-girl* (high frequency), *naughty-dog* (medium) and *naughty-lunch* (low). Each predicate was also matched with three random arguments

	Verb-object				Noun-noun				Adjective-noun			
	Seen		Unseen		Seen		Unseen		Seen		Unseen	
	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ	r	ρ
AltaVista (KL)	**.641**	–	.551	–	**.700**	–	.578	–	**.650**	–	.480	–
Google (KL)	.624	–	.520	–	.692	–	.595	–	.641	–	.473	–
BNC (RASP)	.620	**.614**	.196	.222	.544	.604	.114	.125	.543	**.622**	.135	.102
ROOTH-EM	.455	.487	.479	.520	.503	.491	.586	.625	.514	.463	.395	.355
Padó et al.	.484	.490	.398	.430	.431	.503	.558	.533	.479	.570	.120	.138
LDA	.504	.541	.558	.603	.615	**.641**	.636	.666	.594	.558	.468	.459
ROOTH-LDA	.520	.548	**.564**	**.605**	.607	.622	**.691**	**.722**	.575	.599	**.501**	**.469**
DUAL-LDA	.453	.494	.446	.516	.496	.494	.553	.573	.460	.400	.334	.278

Table 2: Results (Pearson r and Spearman ρ correlations) on Keller and Lapata's (2003) plausibility data

with which it does not co-occur in the BNC (e.g., *naughty-regime, naughty-rival, naughty-protocol*). In this way two datasets (*Seen* and *Unseen*) of 90 items each were assembled for each predicate.

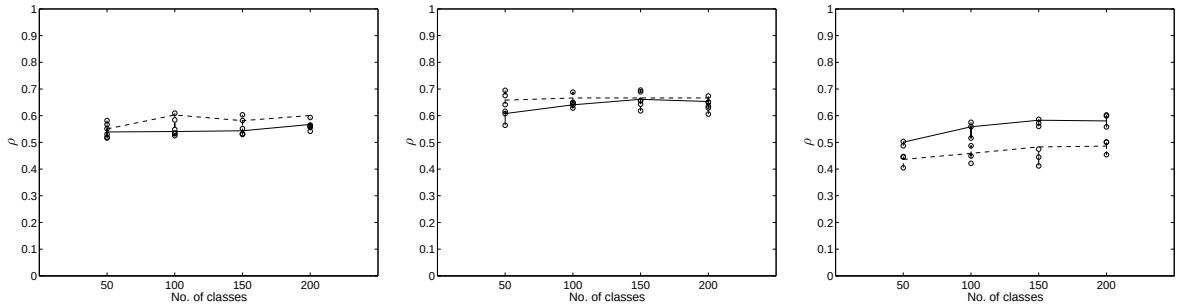
Table 2 presents results for a variety of predictive models – the Web frequencies reported by Keller and Lapata (2003) for two search engines, frequencies from the RASP-parsed BNC,[4] the reimplemented methods of Rooth et al. (1999) and Padó et al. (2007), and the LDA, ROOTH-LDA and DUAL-LDA topic models. Following Keller and Lapata, we report Pearson correlation coefficients between log-transformed predicted frequencies and the gold-standard plausibility scores (which are already log-transformed). We also report Spearman rank correlations except where we do not have the original predictions (the Web count models), for completeness and because the predictions of preference models are may not be log-normally distributed as corpus counts are. Zero values (found only in the BNC frequency predictions) were smoothed by 0.1 to facilitate the log transformation; it seems natural to take a zero prediction as a non-specific prediction of very low plausibility rather than a "missing value" as is done in other work (e.g., Padó et al., 2007).

Despite their structural differences, LDA and ROOTH-LDA perform similarly - indeed, their predictions are highly correlated. ROOTH-LDA scores best overall, outperforming Padó et al.'s (2007) method and ROOTH-EM on every dataset and evaluation measure, and outperforming Keller and Lapata's (2003) Web predictions on every Unseen dataset. LDA also performs consistently well, surpassing ROOTH-EM and Padó et al. on all but one occasion. For frequent predicate-argument pairs (Seen datasets), Web counts are clearly better; however, the BNC counts are unambiguously superior to LDA and ROOTH-LDA (whose predictions are based entirely on the generative model even for observed items) for the Seen verb-object data only. As might be suspected from the mixing problems observed with DUAL-LDA, this model does not perform as well as LDA and ROOTH-LDA, though it does hold its own against the other selectional preference methods.

To identify significant differences between models, we use the statistical test for correlated correlation coefficients proposed by Meng et al. (1992), which is appropriate for correlations that share the same gold standard.[5] For the seen data there are few significant differences: ROOTH-LDA and LDA are significantly better ($p < 0.01$) than Padó et al.'s model for Pearson's r on seen noun-noun data, and ROOTH-LDA is also significantly better ($p < 0.01$) using Spearman's ρ. For the unseen datasets, the BNC frequency predictions are unsurprisingly significantly worse at the $p < 0.01$ level than all smoothing models. LDA and ROOTH-LDA are significantly better ($p < 0.01$) than Padó et al. on every unseen dataset; ROOTH-EM is significantly better ($p < 0.01$) than Padó et al. on Unseen adjectives for both correlations. Meng et al.'s test does not find significant differences between ROOTH-EM and the LDA models despite the latter's clear advantages (a number of conditions do come close). This is because their predictions are highly correlated, which is perhaps

[4]The correlations presented here for BNC counts are notably better than those reported by Keller and Lapata (2003), presumably reflecting our use of full parsing rather than shallow parsing.

[5]We cannot compare our data to Keller and Lapata's Web counts as we do not possess their per-item scores.



(a) Verb-object (b) Noun-noun (c) Adjective-noun

Figure 1: Effect of number of argument classes on Spearman rank correlation with LDA: the solid and dotted lines show the Seen and Unseen datasets respectively; bars show locations of individual samples

unsurprising given that they are structurally similar models trained on the same data. We hypothesise that the main reason for the superior numerical performance of the LDA models over EM is the principled smoothing provided by the use of Dirichlet priors, which has a small but discriminative effect on model predictions. Collating the significance scores, we find that ROOTH-LDA achieves the most positive outcomes, followed by LDA and then by ROOTH-EM. DUAL-LDA is found significantly better than Padó et al.'s model on unseen adjective-noun combinations, and significantly worse than the same model on seen adjective-noun data.

Latent variable models that use EM for inference can be very sensitive to the number of latent variables chosen. For example, the performance of ROOTH-EM worsens quickly if the number of clusters is overestimated; for the Keller and Lapata datasets, settings above 50 classes lead to clear overfitting and a precipitous drop in Pearson correlation scores. On the other hand, Wallach et al. (2009) demonstrate that LDA is relatively insensitive to the choice of topic vocabulary size Z when the α and β hyperparameters are optimised appropriately during estimation. Figure 1 plots the effect of Z on Spearman correlation for the LDA model. In general, Wallach et al.'s finding for document modelling transfers to selectional preference models; within the range $Z = 50$–200 performance remains at a roughly similar level. In fact, we do not find that performance becomes significantly less robust when hyperparameter reestimation is deactiviated; correlation scores simply drop by a small amount (1–2 points), irrespective of the Z chosen. ROOTH-LDA (not graphed) seems slightly more sensitive to Z; this may be because the α parameters in this model operate on the relation level rather than the document level and thus fewer "observations" of class distributions are available when reestimating them.

5.3 Comparison with Bergsma et al. (2008)

As mentioned in Section 2.1, Bergsma et al. (2008) propose a discriminative approach to preference learning. As part of their evaluation, they compare their approach to a number of others, including that of Erk (2007), on a plausibility dataset collected by Holmes et al. (1989). This dataset consists of 16 verbs, each paired with one plausible object (e.g., *write-letter*) and one implausible object (*write-market*). Bergsma et al.'s model, trained on the 3GB AQUAINT corpus, is the only model reported to achieve perfect accuracy on distinguishing plausible from implausible arguments. It would be interesting to do a full comparison that controls for size and type of corpus data; in the meantime, we can report that the LDA and ROOTH-LDA models trained on verb-object observations in the BNC (about 4 times smaller than AQUAINT) also achieve a perfect score on the Holmes et al. data.[6]

6 Conclusions and future work

This paper has demonstrated how Bayesian techniques originally developed for modelling the topical structure of documents can be adapted to learn probabilistic models of selectional preference. These models are especially effective for estimating plausibility of low-frequency items, thus distinguishing rarity from clear implausibility.

The models presented here derive their predictions by modelling predicate-argument plausibility through the intermediary of latent variables. As observed in Section 5.2 this may be a suboptimal

[6] Bergsma et al. report that all plausible pairs were seen in their corpus; three were unseen in ours, as well as 12 of the implausible pairs.

strategy for frequent combinations, where corpus counts are probably reliable and plausibility judgements may be affected by lexical collocation effects. One principled method for folding corpus counts into LDA-like models would be to use hierarchical priors, as in the n-gram topic model of Wallach (2006). Another potential direction for system improvement would be an integration of our generative model with Bergsma et al.'s (2008) discriminative model – this could be done in a number of ways, including using the induced classes of a topic model as features for a discriminative classifier or using the discriminative classifier to produce additional high-quality training data from noisy unparsed text.

Comparison to plausibility judgements gives an intrinsic measure of model quality. As mentioned in the Introduction, selectional preferences have many uses in NLP applications, and it will be interesting to evaluate the utility of Bayesian preference models in contexts such as semantic role labelling or human sentence processing modelling. The probabilistic nature of topic models, coupled with an appropriate probabilistic task model, may facilitate the integration of class induction and task learning in a tight and principled way. We also anticipate that latent variable models will prove effective for learning selectional preferences of semantic predicates (e.g., FrameNet roles) where direct estimation from a large corpus is not a viable option.

Acknowledgements

This work was supported by EPSRC grant EP/G051070/1. I am grateful to Frank Keller and Mirella Lapata for sharing their plausibility data, and to Andreas Vlachos and the anonymous ACL and CoNLL reviewers for their helpful comments.

References

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preferences from unlabeled text. In *Proceedings of EMNLP-08*, Honolulu, HI.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of EMNLP-CoNLL-07*, Prague, Czech Republic.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the ACL-06 Interactive Presentation Sessions*, Sydney, Australia.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of EACL-09*, Athens, Greece.

Lou Burnard, 1995. *Users' Guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Service, Oxford, UK.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS-09*, Vancouver, BC.

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

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL-07*, Prague, Czech Republic.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of ACL-07*, Prague, Czech Republic.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl. 1):5228–5235.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Virginia M. Holmes, Laurie Stowe, and Linda Cupples. 1989. Lexical expectations in parsing complement-verb sentences. *Journal of Memory and Language*, 28(6):668–689.

Frank Keller and Mirella Lapata. 2003. Using the Web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? Identifying perspectives at the document and sentence levels. In *Proceedings of CoNLL-06*, New York, NY.

Xiao-Li Meng, Robert Rosenthal, and Donald B. Rubin. 1992. Comparing correlated correlation coefficients. *Psychological Bulletin*, 111(1):172–175.

Sebastian Padó, Ulrike Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of EMNLP-CoNLL-07*, Prague, Czech Republic.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL-HLT-07*, Rochester, NY.

Keith Rayner, Tessa Warren, Barbara J. Juhasz, and Simon P. Liversedge. 2004. The effect of plausibility on eye movements in reading. *Journal of Experimental Psychology: Learning Memory and Cognition*, 30(6):1290–1301.

Joseph Reisinger and Marius Paşca. 2009. Latent variable models of concept-attribute attachment. In *Proceedings of ACL-IJCNLP-09*, Singapore.

Philip S. Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. Ph.D. thesis, University of Pennsylvania.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A Latent Dirichlet Allocation method for selectional preferences. In *Proceedings of ACL-10*, Uppsala, Sweden.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of ACL-99*, College Park, MD.

Sabine Schulte im Walde, Christian Hying, Christian Scheible, and Helmut Schmid. 2008. Combining EM training and the MDL principle for an automatic verb classification incorporating selectional preferences. In *Proceedings of ACL-08:HLT*, Columbus, OH.

Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Hanna Wallach, David Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of NIPS-09*, Vancouver, BC.

Hanna Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of ICML-06*, Pittsburgh, PA.

Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11:197–225.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of KDD-09*, Paris, France.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Proceedings of ACL-IJCNLP-09*, Singapore.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing topic models for pattern-based semantic class discovery. In *Proceedings of ACL-IJCNLP-09*, Singapore.

Improving the Use of Pseudo-Words for Evaluating Selectional Preferences

Nathanael Chambers and Dan Jurafsky
Department of Computer Science
Stanford University
{natec,jurafsky}@stanford.edu

Abstract

This paper improves the use of pseudo-words as an evaluation framework for selectional preferences. While pseudo-words originally evaluated word sense disambiguation, they are now commonly used to evaluate selectional preferences. A selectional preference model ranks a set of possible arguments for a verb by their semantic fit to the verb. Pseudo-words serve as a proxy evaluation for these decisions. The evaluation takes an argument of a verb like *drive* (e.g. *car*), pairs it with an alternative word (e.g. *car/rock*), and asks a model to identify the original. This paper studies two main aspects of pseudo-word creation that affect performance results. (1) Pseudo-word evaluations often evaluate only a subset of the words. We show that selectional preferences should instead be evaluated on the data in its entirety. (2) Different approaches to selecting partner words can produce overly optimistic evaluations. We offer suggestions to address these factors and present a simple baseline that outperforms the state-of-the-art by 13% absolute on a newspaper domain.

1 Introduction

For many natural language processing (NLP) tasks, particularly those involving meaning, creating labeled test data is difficult or expensive. One way to mitigate this problem is with pseudo-words, a method for automatically creating test corpora without human labeling, originally proposed for word sense disambiguation (Gale et al., 1992; Schutze, 1992). While pseudo-words are now less often used for word sense disambigation, they are a common way to evaluate *selectional preferences*, models that measure the strength of association between a predicate and its argument filler, e.g., that the noun *lunch* is a likely object of *eat*. Selectional preferences are useful for NLP tasks such as parsing and semantic role labeling (Zapirain et al., 2009). Since evaluating them in isolation is difficult without labeled data, pseudo-word evaluations can be an attractive evaluation framework.

Pseudo-word evaluations are currently used to evaluate a variety of language modeling tasks (Erk, 2007; Bergsma et al., 2008). However, *evaluation design varies across research groups*. This paper studies the evaluation itself, showing how choices can lead to overly optimistic results if the evaluation is not designed carefully. We show in this paper that current methods of applying pseudo-words to selectional preferences vary greatly, and suggest improvements.

A pseudo-word is the concatenation of two words (e.g. *house/car*). One word is the original in a document, and the second is the *confounder*. Consider the following example of applying pseudo-words to the selectional restrictions of the verb *focus*:

***Original**: This story focuses on the campaign.*
***Test**: This story/part focuses on the campaign/meeting.*

In the original sentence, *focus* has two arguments: a subject *story* and an object *campaign*. In the test sentence, each argument of the verb is replaced by pseudo-words. A model is evaluated by its success at determining which of the two arguments is the original word.

Two problems exist in the current use of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 445–453,
Uppsala, Sweden, 11-16 July 2010.

pseudo-words to evaluate selectional preferences. First, selectional preferences historically focus on subsets of data such as *unseen* words or words in certain frequency ranges. While work on unseen data is important, evaluating on the entire dataset provides an accurate picture of a model's overall performance. Most other NLP tasks today evaluate *all* test examples in a corpus. We will show that *seen* arguments actually dominate newspaper articles, and thus propose creating test sets that include all verb-argument examples to avoid artificial evaluations.

Second, pseudo-word evaluations vary in how they choose confounders. Previous work has attempted to maintain a similar corpus frequency to the original, but it is not clear how best to do this, nor how it affects the task's difficulty. We argue in favor of using nearest-neighbor frequencies and show how using random confounders produces overly optimistic results.

Finally, we present a surprisingly simple baseline that outperforms the state-of-the-art and is far less memory and computationally intensive. It outperforms current similarity-based approaches by over 13% when the test set includes all of the data. We conclude with a suggested backoff model based on this baseline.

2 History of Pseudo-Word Disambiguation

Pseudo-words were introduced simultaneously by two papers studying statistical approaches to word sense disambiguation (WSD). Schütze (1992) simply called the words, 'artificial ambiguous words', but Gale et al. (1992) proposed the succinct name, *pseudo-word*. Both papers cited the sparsity and difficulty of creating large labeled datasets as the motivation behind pseudo-words. Gale et al. selected unambiguous words from the corpus and paired them with random words from different thesaurus categories. Schütze paired his words with confounders that were 'comparable in frequency' and 'distinct semantically'. Gale et al.'s pseudo-word term continues today, as does Schütze's frequency approach to selecting the confounder.

Pereira et al. (1993) soon followed with a selectional preference proposal that focused on a language model's effectiveness on *unseen* data. The work studied clustering approaches to assist in similarity decisions, predicting which of two verbs was the correct predicate for a given noun object. One verb v was the original from the source document, and the other v' was randomly generated. This was the first use of such verb-noun pairs, as well as the first to test only on *unseen* pairs.

Several papers followed with differing methods of choosing a test pair (v, n) and its confounder v'. Dagan et al. (1999) tested all *unseen* (v, n) occurrences of the most frequent 1000 verbs in his corpus. They then sorted verbs by corpus frequency and chose the neighboring verb v' of v as the confounder to ensure the closest frequency match possible. Rooth et al. (1999) tested 3000 random (v, n) pairs, but required the verbs and nouns to appear between 30 and 3000 times in training. They also chose confounders randomly so that the new pair was *unseen*.

Keller and Lapata (2003) specifically addressed the impact of *unseen* data by using the web to first 'see' the data. They evaluated unseen pseudo-words by attempting to first observe them in a larger corpus (the Web). One modeling difference was to disambiguate the nouns as selectional preferences instead of the verbs. Given a test pair (v, n) and its confounder (v, n'), they used web searches such as "v Det n" to make the decision. Results beat or matched current results at the time. We present a similarly motivated, but new web-based approach later.

Very recent work with pseudo-words (Erk, 2007; Bergsma et al., 2008) further blurs the lines between what is included in training and test data, using frequency-based and semantic-based reasons for deciding what is included. We discuss this further in section 5.

As can be seen, there are two main factors when devising a pseudo-word evaluation for selectional preferences: (1) choosing (v, n) pairs from the test set, and (2) choosing the confounding n' (or v'). The confounder has not been looked at in detail and as best we can tell, these factors have varied significantly. Many times the choices are well motivated based on the paper's goals, but in other cases the motivation is unclear.

3 How Frequent is Unseen Data?

Most NLP tasks evaluate their entire datasets, but as described above, most selectional preference evaluations have focused only on unseen data. This section investigates the extent of unseen examples in a typical training/testing environment

of newspaper articles. The results show that even with a small training size, seen examples dominate the data. We argue that, absent a system's need for specialized performance on unseen data, a representative test set should include the dataset in its entirety.

3.1 Unseen Data Experiment

We use the New York Times (NYT) and Associated Press (APW) sections of the Gigaword Corpus (Graff, 2002), as well as the British National Corpus (BNC) (Burnard, 1995) for our analysis. Parsing and SRL evaluations often focus on newspaper articles and Gigaword is large enough to facilitate analysis over varying amounts of training data. We parsed the data with the Stanford Parser[1] into dependency graphs. Let (v_d, n) be a verb v with grammatical dependency $d \in \{subject, object, prep\}$ filled by noun n. Pairs (v_d, n) are chosen by extracting every such dependency in the graphs, setting the head predicate as v and the head word of the dependent d as n. All prepositions are condensed into *prep*.

We randomly selected documents from the year 2001 in the NYT portion of the corpus as development and test sets. Training data for APW and NYT include all years 1994-2006 (minus NYT development and test documents). We also identified and removed duplicate documents[2]. The BNC in its entirety is also used for training as a single data point. We then record every seen (v_d, n) pair during training that is seen two or more times[3] and then count the number of unseen pairs in the NYT development set (1455 tests).

Figure 1 plots the percentage of unseen arguments against training size when trained on either NYT or APW (the APW portion is smaller in total size, and the smaller BNC is provided for comparison). The first point on each line (the highest points) contains approximately the same number of words as the BNC (100 million). Initially, about one third of the arguments are unseen, but that percentage quickly falls close to 10% as additional training is included. This suggests that an evaluation focusing only on unseen data is not representative, potentially missing up to 90% of the data.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] Any two documents whose first two paragraphs in the corpus files are identical.

[3] Our results are thus conservative, as including all single occurrences would achieve even smaller unseen percentages.

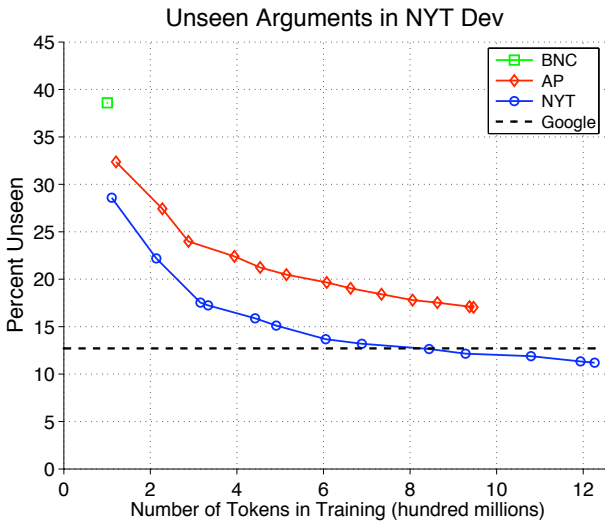


Figure 1: Percentage of NYT development set that is unseen when trained on varying amounts of data. The two lines represent training with NYT or APW data. The APW set is smaller in size from the NYT. The dotted line uses Google n-grams as training. The x-axis represents tokens $\times 10^8$.

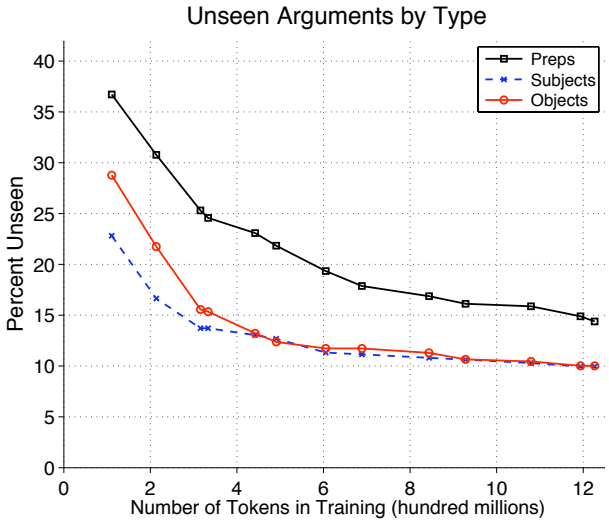


Figure 2: Percentage of subject/object/preposition arguments in the NYT development set that is unseen when trained on varying amounts of NYT data. The x-axis represents tokens $\times 10^8$.

The third line across the bottom of the figure is the number of unseen pairs using Google n-gram data as proxy argument counts. Creating argument counts from n-gram counts is described in detail below in section 5.2. We include these Web counts to illustrate how an openly available source of counts affects unseen arguments. Finally, figure 2 compares which dependency types are seen the least in training. Prepositions have the largest unseen percentage, but not surprisingly, also make up less of the training examples overall.

In order to analyze why pairs are unseen, we analyzed the distribution of *rare* words across unseen and seen examples. To define rare nouns, we order head words by their individual corpus frequencies. A noun is *rare* if it occurs in the lowest 10% of the list. We similarly define *rare verbs* over their ordered frequencies (we count verb lemmas, and do not include the syntactic relations). Corpus counts covered 2 years of the AP section, and we used the development set of the NYT section to extract the seen and unseen pairs. Figure 3 shows the percentage of rare nouns and verbs that occur in unseen and seen pairs. 24.6% of the verbs in unseen pairs are rare, compared to only 4.5% in seen pairs. The distribution of rare nouns is less contrastive: 13.3% vs 8.9%. This suggests that many unseen pairs are unseen mainly because they contain low-frequency verbs, rather than because of containing low-frequency argument heads.

Given the large amount of seen data, we believe evaluations should include all data examples to best represent the corpus. We describe our full evaluation results and include a comparison of different training sizes below.

4 How to Select a Confounder

Given a test set S of pairs $(v_d, n) \in S$, we now address how best to select a confounder n'. Work in WSD has shown that confounder choice can make the pseudo-disambiguation task significantly easier. Gaustad (2001) showed that human-generated pseudo-words are more difficult to classify than random choices. Nakov and Hearst (2003) further illustrated how random confounders are easier to identify than those selected from semantically ambiguous, yet related concepts. Our approach evaluates selectional preferences, not WSD, but our results complement these findings.

We identified three methods of confounder selection based on varying levels of corpus frequency: (1) choose a *random* noun, (2) choose a random noun from a *frequency bucket* similar to the original noun's frequency, and (3) select the *nearest neighbor*, the noun with frequency closest to the original. These methods evaluate the range of choices used in previous work. Our experiments compare the three.

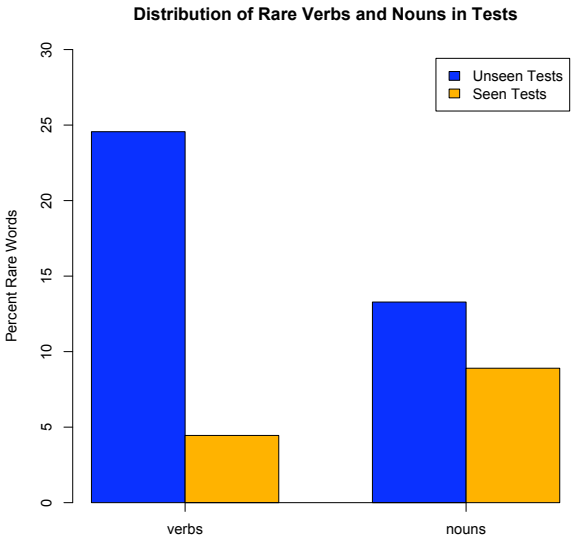


Figure 3: Comparison between seen and unseen tests (verb,relation,noun). 24.6% of unseen tests have rare verbs, compared to just 4.5% in seen tests. The rare nouns are more evenly distributed across the tests.

5 Models

5.1 A New Baseline

The analysis of unseen slots suggests a baseline that is surprisingly obvious, yet to our knowledge, has not yet been evaluated. Part of the reason is that early work in pseudo-word disambiguation explicitly tested only unseen pairs[4]. Our evaluation will include seen data, and since our analysis suggests that up to 90% is seen, a strong baseline should address this seen portion.

[4]Recent work does include some seen data. Bergsma et al. (2008) test pairs that fall below a mutual information threshold (might include some seen pairs), and Erk (2007) selects a subset of roles in FrameNet (Baker et al., 1998) to test and uses all labeled instances within this subset (unclear what portion of subset of data is seen). Neither evaluates all of the seen data, however.

We propose a conditional probability baseline:

$$P(n|v_d) = \begin{cases} \frac{C(v_d,n)}{C(v_d,*)} & \text{if } C(v_d,n) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $C(v_d, n)$ is the number of times the head word n was seen as an argument to the predicate v, and $C(v_d, *)$ is the number of times v_d was seen with any argument. Given a test (v_d, n) and its confounder (v_d, n'), choose n if $P(n|v_d) > P(n'|v_d)$, and n' otherwise. If $P(n|v_d) = P(n'|v_d)$, randomly choose one.

Lapata et al. (1999) showed that corpus frequency and conditional probability correlate with human decisions of adjective-noun plausibility, and Dagan et al. (1999) appear to propose a very similar baseline for verb-noun selectional preferences, but the paper evaluates *unseen* data, and so the conditional probability model is not studied.

We later analyze this baseline against a more complicated smoothing approach.

5.2 A Web Baseline

If conditional probability is a reasonable baseline, better performance may just require more data. Keller and Lapata (2003) proposed using the web for this task, querying for specific phrases like 'Verb Det N' to find syntactic objects. Such a web corpus would be attractive, but we'd like to find subjects and prepositional objects as well as objects, and also ideally we don't want to limit ourselves to patterns. Since parsing the web is unrealistic, a reasonable compromise is to make rough counts when pairs of words occur in close proximity to each other.

Using the Google n-gram corpus, we recorded all verb-noun co-occurrences, defined by appearing in any order in the same n-gram, up to and including 5-grams. For instance, the test pair $(throw_{subject}, ball)$ is considered *seen* if there exists an n-gram such that *throw* and *ball* are both included. We count all such occurrences for all verb-noun pairs. We also avoided over-counting co-occurrences in lower order n-grams that appear again in 4 or 5-grams. This crude method of counting has obvious drawbacks. Subjects are not distinguished from objects and nouns may not be actual arguments of the verb. However, it is a simple baseline to implement with these freely available counts.

Thus, we use conditional probability as defined in the previous section, but define the count $C(v_d, n)$ as the number of times v and n (ignoring d) appear in the same n-gram.

5.3 Smoothing Model

We implemented the current state-of-the-art smoothing model of Erk (2007). The model is based on the idea that the arguments of a particular verb slot tend to be similar to each other. Given two potential arguments for a verb, the correct one should correlate higher with the arguments observed with the verb during training.

Formally, given a verb v and a grammatical dependency d, the score for a noun n is defined:

$$S_{v_d}(n) = \sum_{w \in Seen(v_d)} sim(n,w) * C(v_d, w) \quad (1)$$

where $sim(n, w)$ is a noun-noun similarity score, $Seen(v_d)$ is the set of seen head words filling the slot v_d during training, and $C(v_d, n)$ is the number of times the noun n was seen filling the slot v_d The similarity score $sim(n, w)$ can thus be one of many vector-based similarity metrics[5]. We evaluate both Jaccard and Cosine similarity scores in this paper, but the difference between the two is small.

6 Experiments

Our training data is the NYT section of the Gigaword Corpus, parsed into dependency graphs. We extract all (v_d, n) pairs from the graph, as described in section 3. We randomly chose 9 documents from the year 2001 for a development set, and 41 documents for testing. The test set consisted of 6767 (v_d, n) pairs. All verbs and nouns are stemmed, and the development and test documents were isolated from training.

6.1 Varying Training Size

We repeated the experiments with three different training sizes to analyze the effect data size has on performance:

- **Train x1**: Year 2001 of the NYT portion of the Gigaword Corpus. After removing duplicate documents, it contains approximately 110 million tokens, comparable to the 100 million tokens in the BNC corpus.

[5] A similar type of smoothing was proposed in earlier work by Dagan et al. (1999). A noun is represented by a vector of verb slots and the number of times it is observed filling each slot.

- **Train x2**: Years 2001 and 2002 of the NYT portion of the Gigaword Corpus, containing approximately 225 million tokens.
- **Train x10**: The entire NYT portion of Gigaword (approximately 1.2 billion tokens). It is an order of magnitude larger than *Train x1*.

6.2 Varying the Confounder

We generated three different confounder sets based on word corpus frequency from the 41 test documents. Frequency was determined by counting all tokens with noun POS tags. As motivated in section 4, we use the following approaches:

- **Random**: choose a random confounder from the set of nouns that fall within some broad corpus frequency range. We set our range to eliminate (approximately) the top 100 most frequent nouns, but otherwise arbitrarily set the lower range as previous work seems to do. The final range was $[30, 400000]$.
- **Buckets**: all nouns are bucketed based on their corpus frequencies[6]. Given a test pair (v_d, n), choose the bucket in which n belongs and randomly select a confounder n' from that bucket.
- **Neighbor**: sort all seen nouns by frequency and choose the confounder n' that is the nearest neighbor of n with greater frequency.

6.3 Model Implementation

None of the models can make a decision if they identically score both potential arguments (most often true when both arguments were not seen with the verb in training). As a result, we extend all models to randomly guess (50% performance) on pairs they cannot answer.

The conditional probability is reported as **Baseline**. For the web baseline (reported as **Google**), we stemmed all words in the Google n-grams and counted every verb v and noun n that appear in Gigaword. Given two nouns, the noun with the higher co-occurrence count with the verb is chosen. As with the other models, if the two nouns have the same counts, it randomly guesses.

The smoothing model is named **Erk** in the results with both Jaccard and Cosine as the similarity metric. Due to the large vector representations of the nouns, it is computationally wise to trim their vectors, but also important to do so for best performance. A noun's representative vector consists of verb slots and the number of times the noun was seen in each slot. We removed any verb slot not seen more than x times, where x varied based on all three factors: the dataset, confounder choice, and similarity metric. We optimized x on the development data with a linear search, and used that cutoff on each test. Finally, we trimmed any vectors over 2000 in size to reduce the computational complexity. Removing this strict cutoff appears to have little effect on the results.

Finally, we report backoff scores for Google and Erk. These consist of always choosing the Baseline if it returns an answer (not a guessed unseen answer), and then backing off to the Google/Erk result for Baseline unknowns. These are labeled **Backoff Google** and **Backoff Erk**.

7 Results

Results are given for the two dimensions: *confounder choice* and *training size*. Statistical significance tests were calculated using the approximate randomization test (Yeh, 2000) with 1000 iterations.

Figure 4 shows the performance change over the different confounder methods. *Train x2* was used for training. Each model follows the same progression: it performs extremely well on the random test set, worse on buckets, and the lowest on the nearest neighbor. The conditional probability **Baseline** falls from 91.5 to 79.5, a 12% absolute drop from completely random to neighboring frequency. The **Erk** smoothing model falls 27% from 93.9 to 68.1. The **Google** model generally performs the worst on all sets, but its 74.3% performance with random confounders is significantly better than a 50-50 random choice. This is notable since the Google model only requires n-gram counts to implement. The **Backoff Erk** model is the best, using the **Baseline** for the majority of decisions and backing off to the Erk smoothing model when the Baseline cannot answer.

Figure 5 (shown on the next page) varies the training size. We show results for both Bucket Frequencies and Neighbor Frequencies. The only difference between columns is the amount of training data. As expected, the Baseline improves as the training size is increased. The Erk model, somewhat surprisingly, shows no continual gain with more training data. The Jaccard and Cosine simi-

[6] We used frequency buckets of 4, 10, 25, 200, 1000, >1000. Adding more buckets moves the evaluation closer to *Neighbor*, less is closer to *Random*.

Varying the Confounder Frequency

	Random	Buckets	Neighbor
Baseline	**91.5**	89.1	79.5
Erk-Jaccard	**93.9***	82.7*	68.1*
Erk-Cosine	**91.2**	81.8*	65.3*
Google	**74.3***	70.4*	59.4*
Backoff Erk	**96.6***	91.8*	80.8*
Backoff Goog	**92.7†**	89.7	79.8

Figure 4: Trained on two years of NYT data (Train x2). Accuracy of the models on the same NYT test documents, but with three different ways of choosing the confounders. * indicates statistical significance with the column's Baseline at the $p < 0.01$ level, † at $p < 0.05$. Random is overly optimistic, reporting performance far above more conservative (selective) confounder choices.

Baseline Details

	Train	Train x2	Train x10
Precision	96.1	95.5*	95.0†
Accuracy	78.2	82.0*	88.1*
Accuracy +50%	87.5	89.1*	91.7*

Figure 6: Results from the *buckets* confounder test set. Baseline precision, accuracy (the same as recall), and accuracy when you randomly guess the tests that Baseline does not answer. All numbers are statistically significant * with p-value < 0.01 from the number to their left.

larity scores perform similarly in their model. The Baseline achieves the highest accuracies (91.7% and 81.2%) with *Train x10*, outperforming the best Erk model by 5.2% and 13.1% absolute on buckets and nearest neighbor respectively. The backoff models improve the baseline by just under 1%. The Google n-gram backoff model is almost as good as backing off to the Erk smoothing model.

Finally, figure 6 shows the Baseline's precision and overall accuracy. Accuracy is the same as recall when the model does not guess between pseudo words that have the same conditional probabilities. *Accuracy +50%* (the full Baseline in all other figures) shows the gain from randomly choosing one of the two words when uncertain. Precision is extremely high.

8 Discussion

Confounder Choice: Performance is strongly influenced by the method used when choosing confounders. This is consistent with findings for WSD that corpus frequency choices alter the task (Gaustad, 2001; Nakov and Hearst, 2003). Our results show the gradation of performance as one moves across the spectrum from completely random to closest in frequency. The Erk model dropped 27%, Google 15%, and our baseline 12%. The overly optimistic performance on random data suggests using the nearest neighbor approach for experiments. Nearest neighbor avoids evaluating on 'easy' datasets, and our baseline (at 79.5%) still provides room for improvement. But perhaps just as important, the nearest neighbor approach facilitates the most reproducibile results in experiments since there is little ambiguity in how the confounder is selected.

Realistic Confounders: Despite its overoptimism, the random approach to confounder selection may be the correct approach in some circumstances. For some tasks that need selectional preferences, random confounders may be more realistic. It's possible, for example, that the options in a PP-attachment task might be distributed more like the random rather than nearest neighbor models. In any case, this is difficult to decide without a specific application in mind. Absent such specific motiviation, a nearest neighbor approach is the most conservative, and has the advantage of creating a reproducible experiment, whereas random choice can vary across design.

Training Size: Training data improves the conditional probability baseline, but does not help the smoothing model. Figure 5 shows a lack of improvement across training sizes for both jaccard and cosine implementations of the Erk model. The *Train x1* size is approximately the same size used in Erk (2007), although on a different corpus. We optimized argument cutoffs for each training size, but the model still appears to suffer from additional noise that the conditional probability baseline does not. This may suggest that observing a test argument with a verb in training is more reliable than a smoothing model that compares all training arguments against that test example.

High Precision Baseline: Our conditional probability baseline is very precise. It outperforms the smoothed similarity based **Erk** model and gives high results across tests. The only combination when Erk is better is when the training data includes just one year (one twelfth of the NYT section) and the confounder is chosen com-

Varying the Training Size

	Bucket Frequency			Neighbor Frequency		
	Train x1	Train x2	Train x10	Train x1	Train x2	Train x10
Baseline	87.5	89.1	**91.7**	78.4	79.5	**81.2**
Erk-Jaccard	86.5*	82.7*	83.1*	66.8*	68.1*	65.5*
Erk-Cosine	82.1*	81.8*	81.1*	66.1*	65.3*	65.7*
Google	-	-	70.4*	-	-	59.4*
Backoff Erk	92.6*	91.8*	92.6*	79.4*	80.8*	**81.7***
Backoff Google	88.6	89.7	91.9†	78.7	79.8	81.2

Figure 5: Accuracy of varying NYT training sizes. The left and right tables represent two confounder choices: choose the confounder with frequency buckets, and choose by nearest frequency neighbor. Trainx1 starts with year 2001 of NYT data, Trainx2 doubles the size, and Trainx10 is 10 times larger. * indicates statistical significance with the column's Baseline at the $p < 0.01$ level, † at $p < 0.05$.

pletely randomly. These results appear consistent with Erk (2007) because that work used the BNC corpus (the same size as one year of our data) and Erk chose confounders randomly within a broad frequency range. Our reported results include every (v_d, n) in the data, not a subset of particular semantic roles. Our reported 93.9% for Erk-Jaccard is also significantly higher than their reported 81.4%, but this could be due to the random choices we made for confounders, or most likely corpus differences between Gigaword and the subset of FrameNet they evaluated.

Ultimately we have found that complex models for selectional preferences may not be necessary, depending on the task. The higher computational needs of smoothing approaches are best for backing off when unseen data is encountered. Conditional probability is the best choice for seen examples. Further, analysis of the data shows that as more training data is made available, the seen examples make up a much larger portion of the test data. Conditional probability is thus a very strong starting point if selectional preferences are an internal piece to a larger application, such as semantic role labeling or parsing.

Perhaps most important, these results illustrate the disparity in performance that can come about when designing a pseudo-word disambiguation evaluation. It is crucially important to be clear during evaluations about how the confounder was generated. We suggest the approach of sorting nouns by frequency and using a neighbor as the confounder. This will also help avoid evaluations that produce overly optimistic results.

9 Conclusion

Current performance on various natural language tasks is being judged and published based on pseudo-word evaluations. It is thus important to have a clear understanding of the evaluation's characteristics. We have shown that the evaluation is strongly affected by confounder choice, suggesting a nearest frequency neighbor approach to provide the most reproducible performance and avoid overly optimistic results. We have shown that evaluating entire documents instead of subsets of the data produces vastly different results. We presented a conditional probability baseline that is both novel to the pseudo-word disambiguation task and strongly outperforms state-of-the-art models on entire documents. We hope this provides a new reference point to the pseudo-word disambiguation task, and enables selectional preference models whose performance on the task similarly transfers to larger NLP applications.

Acknowledgments

This work was supported by the National Science Foundation IIS-0811974, and the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, ndings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reect the view of the AFRL. Thanks to Sebastian Padó, the Stanford NLP Group, and the anonymous reviewers for very helpful suggestions.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *ACL-98*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Empirical Methods in Natural Language Processing*, pages 59–68, Honolulu, Hawaii.

Lou Burnard. 1995. *User Reference Guide for the British National Corpus*. Oxford University Press, Oxford.

Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1):43–69.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

Tanja Gaustad. 2001. Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *39th Annual Meeting of the Association for Computational Linguistics - Student Research Workshop*.

David Graff. 2002. English Gigaword. *Linguistic Data Consortium*.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Maria Lapata, Scott McDonald, and Frank Keller. 1999. Determinants of adjective-noun plausibility. In *European Chapter of the Association for Computational Linguistics (EACL)*.

Preslav I. Nakov and Marti A. Hearst. 2003. Category-based pseudowords. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 67–69, Edmonton, Canada.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Hinrich Schutze. 1992. Context space. In *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 113–120.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *International Conference on Computational Linguistics (COLING)*.

Beat Zapirain, Eneko Agirre, and Llus Mrquez. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*, Singapore.

Syntax-to-Morphology Mapping in Factored Phrase-Based Statistical Machine Translation from English to Turkish

Reyyan Yeniterzi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
reyyan@cs.cmu.edu

Kemal Oflazer
Computer Science
Carnegie Mellon University-Qatar
PO Box 24866, Doha, Qatar
ko@cs.cmu.edu

Abstract

We present a novel scheme to apply factored phrase-based SMT to a language pair with very disparate morphological structures. Our approach relies on syntactic analysis on the source side (English) and then encodes a wide variety of local and non-local syntactic structures as *complex structural tags* which appear as additional factors in the training data. On the target side (Turkish), we only perform morphological analysis and disambiguation but treat the complete complex morphological tag as a factor, instead of separating morphemes. We incrementally explore capturing various syntactic substructures as complex tags on the English side, and evaluate how our translations improve in BLEU scores. Our maximal set of source and target side transformations, coupled with some additional techniques, provide an 39% relative improvement from a baseline 17.08 to 23.78 BLEU, all averaged over 10 training and test sets. Now that the syntactic analysis on the English side is available, we also experiment with more long distance constituent reordering to bring the English constituent order close to Turkish, but find that these transformations do not provide any additional consistent tangible gains when averaged over the 10 sets.

1 Introduction

Statistical machine translation into a morphologically complex language such as Turkish, Finnish or Arabic, involves the generation of target words with the proper morphology, in addition to properly ordering the target words. Earlier work on translation from English to Turkish (Oflazer and Durgar-El-Kahlout, 2007; Oflazer, 2008; Durgar-El-Kahlout and Oflazer, 2010) has used an approach which relied on identifying the contextually correct parts-of-speech, roots and any morphemes on the English side, and the complete sequence of roots and overt derivational and inflectional morphemes for each word on the Turkish side. Once these were identified as separate tokens, they were then used as "words" in a standard phrase-based framework (Koehn et al., 2003). They have reported that, given the typical complexity of Turkish words, there was a substantial percentage of words whose morphological structure was incorrect: either the morphemes were not applicable for the part-of-speech category of the root word selected, or the morphemes were in the wrong order. The main reason given for these problems was that the same statistical translation, reordering and language modeling mechanisms were being employed to *both* determine the morphological structure of the words *and*, at the same time, get the global order of the words correct. Even though a significant improvement of a standard word-based baseline was achieved, further analysis hinted at a direction where morphology and syntax on the Turkish side had to be dealt with using separate mechanisms.

Motivated by the observation that many local and some nonlocal syntactic structures in English essentially map to morphologically complex words in Turkish, we present a radically different approach which does not segment Turkish words into morphemes, but uses a representation equivalent to the full word form. On the English side, we rely on a full syntactic analysis using a dependency parser. This analysis then lets us abstract and encode many local and some nonlocal syntactic structures as complex tags (dynamically, as opposed to the static complex tags as proposed by Birch et *al.* (2007) and Hassan et *al.* (2007)). Thus

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 454–464,
Uppsala, Sweden, 11-16 July 2010.

we can bring the representation of English syntax closer to the Turkish morphosyntax.

Such an approach enables the following: (i) Driven by the pattern of morphological structures of full word forms on the Turkish side represented as root words and complex tags, we can identify and reorganize phrases on the English side, to "align" English syntax to Turkish morphology wherever possible. (ii) Continuous and discontinuous variants of certain (syntactic) phrases can be conflated during the SMT phrase extraction process. (iii) The length of the English sentences can be dramatically reduced, as most function words encoding syntax are now abstracted into complex tags. (iv) The representation of both the source and the target sides of the parallel corpus can now be mostly normalized. *This facilitates the use of factored phrase-based translation that was not previously applicable due to the morphological complexity on the target side and mismatch between source and target morphologies.*

We find that with the full set of syntax-to-morphology transformations and some additional techniques we can get about 39% relative improvement in BLEU scores over a word-based baseline and about 28% improvement of a factored baseline, all experiments being done over 10 training and test sets. We also find that further constituent reordering taking advantage of the syntactic analysis of the source side, does not provide tangible improvements when averaged over the 10 data sets.

This paper is organized as follows: Section 2 presents the basic idea behind syntax-to-morphology alignment. Section 3 describes our experimental set-up and presents results from a sequence of incremental syntax-to-morphology transformations, and additional techniques. Section 4 summarizes our constituent reordering experiments and their results. Section 5 presents a review of related work and situates our approach.

We assume that the reader is familiar with the basics of phrase-based statistical machine translation (Koehn et al., 2003) and factored statistical machine translation (Koehn and Hoang, 2007).

2 Syntax-to-Morphology Mapping

In this section, we describe how we map between certain source language syntactic structures and target words with complex morphological structures. At the top of Figure 1, we see a pair of (syntactic) phrases, where we have (positionally) aligned the words that should be translated to each other. We can note that the function words *on* and

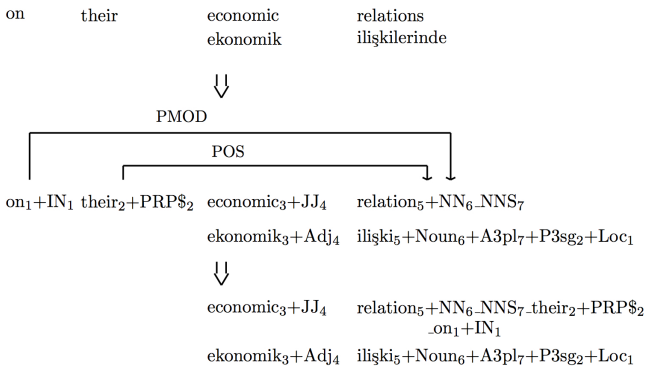


Figure 1: Transformation of an English prepositional phrase

their are not really aligned to any of the Turkish words as they really correspond to two of the morphemes of the last Turkish word.

When we tag and syntactically analyze the English side into dependency relations, and morphologically analyze and disambiguate the Turkish phrase, we get the representation in the middle of Figure 1, where we have co-indexed components that should map to each other, and some of the syntactic relations that the function words are involved in are marked with dependency links.[1]

The basic idea in our approach is to take various function words on the English side, whose syntactic relationships are identified by the parser, and then package them as complex tags on the related content words. So, in this example, if we move the first two function words from the English side and attach as *syntactic tags* to the word they are in dependency relation with, we get the aligned representation at the bottom of Figure 1.[2,3] Here we can note that all root words and tags that correspond to each other are nicely structured and are in the same relative order. In fact, we can treat each token as being composed of two factors: the roots and the accompanying tags. The tags on the Turkish side encode morphosyntactic information encoded in the morphology of the words, *while the*

[1]The meanings of various tags are as follows: Dependency Labels: **PMOD** - Preposition Modifier; **POS** - Possessive. Part-of-Speech Tags for the English words: **+IN** - Preposition; **+PRP$** - Possessive Pronoun; **+JJ** - Adjective; **+NN** - Noun; **+NNS** - Plural Noun. Morphological Feature Tags in the Turkish Sentence: **+A3pl** - 3rd person plural; **+P3sg** - 3rd person singular possessive; **+Loc** - Locative case. Note that we mark an English plural noun as **+NN_NNS** to indicate that the root is a noun and there is a plural morpheme on it. Note also that *economic* is also related to *relations* but we are not interested in such content words and their relations.

[2]We use _ to prefix such syntactic tags on the English side.

[3]The order is important in that we would like to attach the same sequence of function words in the same order so that the resulting tags on the English side are the same.

(complex) tags on the English side encode local (and sometimes, non-local) syntactic information. Furthermore, we can see that before the transformations, the English side has 4 words, while afterwards it has only 2 words. We find (and elaborate later) that this reduction in the English side of the training corpus, in general, is about 30%, and is correlated with improved BLEU scores. We believe the removal of many function words and their folding into complex tags (which do not get involved in GIZA++ alignment – we only align the root words) seems to improve alignment as there are less number of "words" to worry about during that process.[4]

Another interesting side effect of this representation is the following. As the complex syntactic tags on the English side are based on syntactic relations and not necessarily positional proximity, the tag for *relations* in a phrase like *in their cultural, historical and economic relations* would be exactly the same as above. Thus phrase extraction algorithms can conflate all constructs like *in their ... economic relations* as one phrase, regardless of the intervening modifiers, assuming that parser does its job properly.

Not all cases can be captured as cleanly as the example above, but most transformations capture local and nonlocal syntax involving many function words and then encode syntax with complex tags resembling full morphological tags on the Turkish side. *These transformations, however, are not meant to perform sentence level constituent reordering on the English side.* We explore these later.

We developed set of about 20 linguistically-motivated syntax-to-morphology transformations which had variants parameterized depending on what, for instance, the preposition or the adverbial was, and how they map to morphological structure on the Turkish side. For instance, one general rule handles cases like *while ... verb* and *if ... verb* etc., mapping these to appropriate complex tags. It is also possible that multiple transformations can apply to generate a single English complex tag: a portion of the tag can come from a verb complex transformation, and another from an adverbial phrase transformation involving a marked such as *while*. Our transformations handle the following cases:

- Prepositions attach to the head-word of their complement noun phrase as a component in its complex tag.
- Possessive pronouns attach to the head-word they specify.
- The possessive markers following a noun (separated by the tokenizer) attached to the noun.
- Auxiliary verbs and negation markers attach to the lexical verb that they form a verb complex with.
- Modals attach to the lexical verb they modify.
- Forms of *be* used as predicates with adjectival or nominal dependents attach to the dependent.
- Forms of *be* or *have* used to form passive voice with past participle verbs, and forms of *be* used with *-ing* verbs to form present continuous verbs, attach to the verb.
- Various adverbial clauses formed with *if*, *while*, *when*, etc., are reorganized so that these markers attach to the head verb of the clause.

As stated earlier, these rules are linguistically motivated and are based on the morphological structure of the *target* language words. Hence for different target languages these rules will be different. The rules recognize various local and nonlocal syntactic structures in the source side parse tree that correspond to complex morphological of target words and then remove source function words folding them into complex tags. For instance, the transformations in Figure 1 are handled by scripts that process Malt Parser's dependency structure output and that essentially implement the following sequence of rules expressed as pseudo code:

```
1) if  (<Y>+PRP$ POS <Z>+NN<TAG>)
   then {
        APPEND <Y>+PRP$ TO <Z>+NN<TAG>
        REMOVE <Y>+PRP$
     }

2) if (<X>+IN  PMOD <Z>+NN<TAG>)
   then {
        APPEND <X>+IN TO <Z>+NN<TAG>
        REMOVE <X>+IN
     }
```

Here <X>, <Y> and <Z> can be considered as Prolog like-variables that bind to patterns (mostly root words), and the conditions check for specified dependency relations (e.g., PMOD) between the left and the right sides. When the condition is satisfied, then the part matching the function word is removed and its syntactic information is appended to form the complex tag on the noun (<TAG> would either match null string or any previously appended function word markers.)[5]

[4] Fraser (2009) uses the first four letters of German words after morphological stripping and compound decomposition to help with alignment in German to English and reverse translation.

[5] We outline two additional rules later when we see a more complex example in Figure 2.

There are several other rules that handle more mundane cases of date and time constructions (for which, the part of the date construct which the parser attaches a preposition, is usually different than the part on the Turkish side that gets inflected with case markers, and these have to be reconciled by overriding the parser output.)

The next section presents an example of a sentence with multiple transformations applied, after discussing the preprocessing steps.

3 Experimental Setup and Results

3.1 Data Preparation

We worked on an English-Turkish parallel corpus which consists of approximately 50K sentences with an average of 23 words in English sentences and 18 words in Turkish sentences. This is the same parallel data that has been used in earlier SMT work on Turkish (Durgar-El-Kahlout and Oflazer, 2010). Let's assume we have the following pair of parallel sentences:

E: if a request is made orally the authority must make a record of it
T: istek sözlü olarak yapılmışsa yetkili makam bunu kaydetmelidir

On the English side of the data, we use the Stanford Log-Linear Tagger (Toutanova et al., 2003), to tag the text with Penn Treebank Tagset. On the Turkish side, we perform a full morphological analysis, (Oflazer, 1994), and morphological disambiguation (Yuret and Türe, 2006) to select the contextually salient interpretation of words. We then remove any morphological features that are not explicitly marked by an *overt morpheme*.[6] So for both sides we get,

E: if+IN a+DT request+NN is+VBZ made+VBN orally+RB the+DT authority+NN must+MD make+VB a+DT record+NN of+IN it+PRP
T: istek+Noun sözlü+Adj olarak+Verb+ByDoingSo yap+Verb+Pass+Narr+Cond yetkili+Adj makam+Noun bu+Pron+Acc kaydet+Verb+Neces+Cop

Finally we parse the English sentences using MaltParser (Nivre et al., 2007), which gives us labeled dependency parses. On the output of the parser, we make one more transformation. We replace each word with its root, and possibly add an additional tag for any inflectional information conveyed by overt morphemes or exceptional forms. This is done by running the TreeTagger (Schmid, 1994) on the English side which provides the roots in addition to the tags, and then carrying over this information to the parser output. For example, `is` is tagged as `be+VB_VBZ`, `made` is tagged as `make+VB_VBN`, and a word like `books` is tagged as `book+NN_NNS` (and not as `books+NNS`). On the Turkish side, each marker with a preceding + is a morphological feature. The first marker is the part-of-speech tag of the root and the remainder are the overt inflectional and derivational markers of the word. For example, the analysis `kitap+Noun+P2pl+A3pl+Gen` for a word like `kitap-lar-ınız-ın`[7] (*of your books*) represents the root `kitap` (*book*), a `Noun`, with third person plural agreement `A3pl`, second person plural possessive agreement, `P2pl` and genitive case `Gen`.

The sentence representations in the middle part of Figure 2 show these sentences with some of the dependency relations (relevant to our transformations) extracted by the parser, explicitly marked as labeled links. The representation at the bottom of this figure (except for the co-indexation markers) corresponds to the final transformed form of the parallel training and test data. The co-indexation is meant to show which root words on one side map to which on the other side. Ultimately we would want the alignment process to uncover the *root word alignments* indicated here. *We can also note that the initial form of the English sentence has 14 words and the final form after transformations, has 7 words (with complex tags).*[8]

3.2 Experiments

We evaluated the impact of the transformations in factored phrase-based SMT with an English-Turkish data set which consists of 52712 parallel sentences. In order to have more confidence in the impact of our transformations, we randomly generated 10 training, test and tune set combinations. For each combination, the latter two were 1000 sentences each and the remaining 50712 sentences were used as training sets.[9,10]

We performed our experiments with the Moses toolkit (Koehn et al., 2007). In order to encourage long distance reordering in the decoder, we used a distortion limit of -1 and a distortion weight of

[6] For example, the morphological analyzer outputs `+A3sg` to mark a singular noun, if there is no explicit plural morpheme. Such markers are removed.

[7] – shows surface morpheme boundaries.

[8] We could give two more examples of rules to process the if-clause in the example in Figure 2. These rules would be applied sequentially: The first rule recognizes the passive construction mediated by `be+VB<AGR>` forming a verb complex (`VC`) with `<Y>+VB_VBN` and appends the former to the complex tag on the latter and then deletes the former token. The second rule then recognizes `<X>+IN` relating to `<Y>+VB<TAGS>`with `VMOD` and appends the former to the complex tag on the latter and then deletes the former token.

[9] The tune set was not used in this work but reserved for future work so that meaningful comparisons could be made.

[10] It is possible that the 10 test sets are not mutually exclusive.

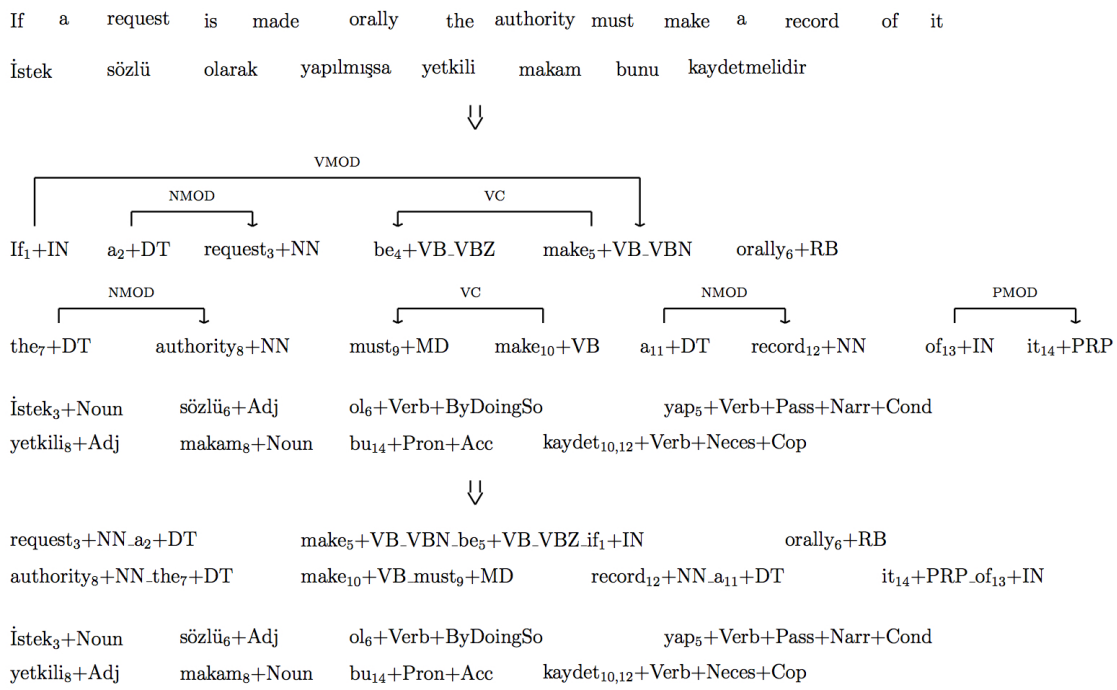


Figure 2: An English-Turkish sentence pair with multiple transformations applied

0.1.[11] We did not use MERT to further optimize our model.[12]

For evaluation, we used the BLEU metric (Papineni et al., 2001). Each experiment was repeated over the 10 data sets. Wherever meaningful, we report the average BLEU scores over 10 data sets along with the maximum and minimum values and the standard deviation.

[11]These allow and do not penalize unlimited distortions.

[12]The experience with MERT for this language pair has not been very positive. Earlier work on Turkish indicates that starting with default Moses parameters and applying MERT to the resulting model does not even come close to the performance of the model with those two specific parameters set as such (*distortion limit -1* and *distortion weight 0.1*), most likely because the default parameters do not encourage the range of distortions that are needed to deal with the constituent order differences. Earlier work on Turkish also shows that even when the *weight-d* parameter is initialized with this specific value, the space explored for distortion weight and other parameters do not produce any improvements on the test set, even though MERT claims there are improvements on the tune set.

The other practical reasons for not using MERT were the following: at the time we performed this work, the discussion thread at `http://www.mail-archive.com/moses-support@mit.edu/msg01012.html` indicated that MERT was not tested on multiple factors. The discussion thread at `http://www.mail-archive.com/moses-support@mit.edu/msg00262.html` claimed that MERT does not help very much with factored models. With these observations, we opted not to experiment with MERT with the multiple factor approach we employed, given that it would be risky and time consuming to run MERT needed for 10 different models and then not necessarily see any (consistent) improvements. MERT however is orthogonal to the improvements we achieve here and can always be applied on top of the best model we get.

3.2.1 The Baseline Systems

As a baseline system, we built a standard phrase-based system, using the surface forms of the words without any transformations, and with a 3-gram LM in the decoder. We also built a second baseline system with a factored model. Instead of using just the surface form of the word, we included the root, part-of-speech and morphological tag information into the corpus as additional factors alongside the surface form.[13] Thus, a token is represented with three factors as `Surface|Root|Tags` where `Tags` are complex tags on the English side, and morphological tags on the Turkish side.[14]

Moses lets word alignment to align over any of the factors. We aligned our training sets using only the root factor to conflate statistics from different forms of the same root. The rest of the factors are then automatically assumed to be aligned, based on the root alignment. Furthermore, in factored models, we can employ different language models for different factors. For the initial set of experiments we used 3-gram LMs for all the factors.

For factored decoding, we employed a model whereby we let the decoder translate a surface form directly, but if/when that fails, the decoder can back-off with a generation model that builds a target word from independent translations of the root and tags.

[13]In Moses, factors are separated by a '|' symbol.

[14]Concatenating `Root` and `Tags` gives the `Surface` form, in that the surface is unique given this concatenation.

The results of our baseline models are given in top two rows of Table 1. As expected, the word-based baseline performs worse than the factored baseline. We believe that the use of multiple language models (some much less sparse than the surface LM) in the factored baseline is the main reason for the improvement.

3.2.2 Applying Syntax-to-Morphology Mapping Transformations

To gauge the effects of transformations separately, we first performed them in batches on the English side. These batches were (i) transformations involving nouns and adjectives (*Noun+Adj*), (ii) transformations involving verbs (*Verb*), (iii) transformations involving adverbs (*Adv*), and (iv) transformations involving verbs and adverbs (*Verb+Adv*).

We also performed one set of transformations on the Turkish side. In general, English prepositions translate as case markers on Turkish nouns. However, there are quite a number of lexical *postpositions* in Turkish which also correspond to English prepositions. To normalize these with the handling of case-markers, we treated these postpositions as if they were case-markers and attached them to the immediately preceding noun, and then aligned the resulting training data (*PostP*).[15]

The results of these experiments are presented in Table 1. We can observe that the combined syntax-to-morphology transformations on the source side provide a substantial improvement by themselves and a simple target side transformation on top of those provides a further boost to 21.96 BLEU which represents a 28.57% relative improvement over the word-based baseline and a 18.00% relative improvement over the factored baseline.

Experiment	Ave.	STD	Max.	Min.
Baseline	17.08	0.60	17.99	15.97
Factored Baseline	**18.61**	**0.76**	**19.41**	**16.80**
Noun+Adj	21.33	0.62	22.27	20.05
Verb	19.41	0.62	20.19	17.99
Adv	18.62	0.58	19.24	17.30
Verb+Adv	19.42	0.59	20.17	18.13
Noun+Adj +Verb+Adv	**21.67**	**0.72**	**22.66**	**20.38**
Noun+Adj+Verb +Adv+PostP	**21.96**	**0.72**	**22.91**	**20.67**

Table 1: BLEU scores for a variety of transformation combinations

We can see that every transformation improves the baseline system and the highest performance is attained when all transformations are performed. However when we take a closer look at the individual transformations performed on English side, we observe that not all of them have the same effect. While *Noun+Adj* transformations give us an increase of 2.73 BLEU points, *Verbs* improve the result by only 0.8 points and improvement with *Adverbs* is even lower. To understand why we get such a difference, we investigated the correlation of the decrease in the number of tokens on both sides of the parallel data, with the change in BLEU scores. The graph in Figure 3 plots the BLEU scores and the number of tokens in the two sides of the training data as the data is modified with transformations. We can see that as the number of tokens in English decrease, the BLEU score increases. In order to measure the relationship between these two variables statistically, we performed a correlation analysis and found that there is a strong negative correlation of -0.99 between the BLEU score and the number of English tokens. We can also note that the largest reduction in the number of tokens comes with the application of the *Noun+Adj* transformations, which correlates with the largest increase in BLEU score.

It is also interesting to look at the n-gram precision components of the BLEU scores (again averaged). In Table 2, we list these for words (actual BLEU), roots (BLEU-R) to see how effective we are in getting the root words right, and morphological tags, (BLEU-M), to see how effective we are in getting just the morphosyntax right. It

		1-gr.	2-gr.	3-gr.	4-gr.
BLEU	21.96	55.73	27.86	16.61	10.68
BLEU-R	27.63	68.60	35.49	21.08	13.47
BLEU-M	27.93	67.41	37.27	21.40	13.41

Table 2: Details of Word, Root and Morphology BLEU Scores

seems we are getting almost 69% of the root words and 68% of the morphological tags correct, but not necessarily getting the combination equally as good, since only about 56% of the full word forms are correct. One possible way to address is to use longer distance constraints on the morphological tag factors, to see if we can select them better.

3.2.3 Experiments with higher-order language models

Factored phrase-based SMT allows the use of multiple language models for the target side, for different factors during decoding. Since the number of possible distinct morphological tags (the morphological tag vocabulary size) in our training data

[15]Note than in this case, the translations would be generated in the same format, but we then split such postpositions from the words they are attached to, during decoding, and *then* evaluate the BLEU score.

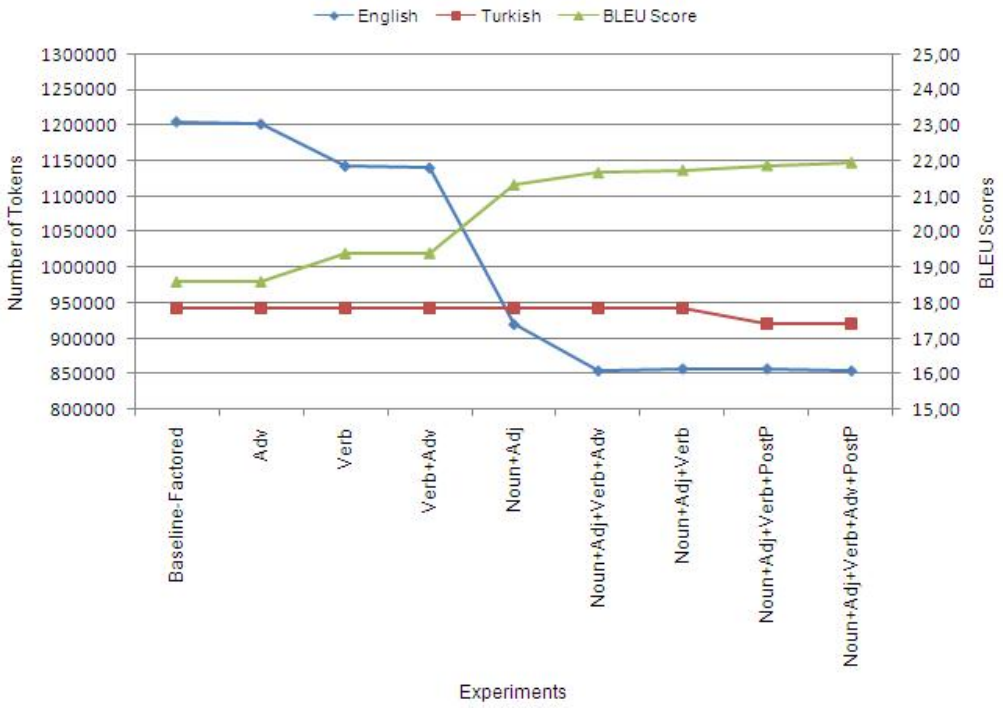


Figure 3: BLEU scores vs number of tokens in the training sets

(about 3700) is small compared to distinct number of surface forms (about 52K) and distinct roots (about 15K including numbers), it makes sense to investigate the contribution of higher order n-gram language models for the morphological tag factor on the target side, to see if we can address the observation in the previous section.

Using the data transformed with *Noun+Adj+Verb+Adv+PostP* transformations which previously gave us the best results overall, we experimented with using higher order models (4-grams to 9-grams) during decoding, for the morphological tag factor models, keeping the surface and root models at 3-gram. We observed that for all the 10 data sets, the improvements were consistent for up to 8-gram. The BLEU with the 8-gram *for only the morphological tag factor* averaged over the 10 data sets was **22.61** (max: 23.66, min: 21.37, std: 0.72) compared to the 21.96 in Table 1. Using a 4-gram *root* LM, considerably less sparse than word forms but more sparse that tags, we get a BLEU score of **22.80** (max: 24.07, min: 21.57, std: 0.85). The details of the various BLEU scores are shown in the two halves of Table 3. It seems that larger n-gram LMs contribute to the larger n-gram precisions contributing to the BLEU but not to the unigram precision.

3-gram root LM		**1-gr.**	**2-gr.**	**3-gr.**	**4-gr.**
BLEU	22.61	55.85	28.21	17.16	11.36
BLEU-R	28.21	68.67	35.80	21.55	14.07
BLEU-M	28.68	67.50	37.59	22.02	14.22
4-gram root LM		**1-gr.**	**2-gr.**	**3-gr.**	**4-gr.**
BLEU	22.80	55.85	28.39	17.34	11.54
BLEU-R	28.48	68.68	35.97	21.79	14.35
BLEU-M	28.82	67.49	37.63	22.17	14.40

Table 3: Details of Word, Root and Morphology BLEU Scores, with 8-gram tag LM and 3/4-gram root LMs

3.2.4 Augmenting the Training Data

In order to alleviate the lack of large scale parallel corpora for the English–Turkish language pair, we experimented with augmenting the training data with reliable phrase pairs obtained from a previous alignment. Phrase table entries for the surface factors produced by Moses after it does an alignment on the roots, contain the English (e) and Turkish (t) parts of a pair of aligned phrases, and the probabilities, $p(e|t)$, the conditional probability that the English phrase is e given that the Turkish phrase is t, and $p(t|e)$, the conditional probability that the Turkish phrase is t given the English phrase is e. Among these phrase table entries, those with $p(e|t) \approx p(t|e)$ and $p(t|e) + p(e|t)$ larger than some threshold, can be considered as reliable mutual translations, in that they mostly translate to each other and not much to others. We extracted

from the phrase table those phrases with $0.9 \leq p(e|t)/p(t|e) \leq 1.1$ and $p(t|e) + p(e|t) \geq 1.5$ and added them to the training data to further bias the alignment process. The resulting BLEU score was **23.78** averaged over 10 data sets (max: 24.52, min: 22.25, std: 0.71).[16]

4 Experiments with Constituent Reordering

The transformations in the previous section *do not perform any constituent level reordering*, but rather *eliminate* certain English function words as tokens in the text and fold them into complex syntactic tags. That is, no transformations reorder the English SVO order to Turkish SOV,[17] for instance, or move postnominal prepositional phrase modifiers in English, to prenominal phrasal modifiers in Turkish. Now that we have the parses of the English side, we have also investigated a more comprehensive set of reordering transformations which perform the following constituent reorderings to bring English constituent order more in line with the Turkish constitent order at the top and embedded phrase levels:

- Object reordering (*ObjR*), in which the objects and their dependents are moved in front of the verb.
- Adverbial phrase reordering (*AdvR*), which involve moving post-verbal adverbial phrases in front of the verb.
- Passive sentence agent reordering (*PassAgR*), in which any post-verbal agents marked by *by*, are moved in front of the verb.
- Subordinate clause reordering (*SubCR*) which involve moving postnominal relative clauses or prepositional phrase modifers in front of any modifiers of the head noun. Similarly any prepositional phrases attached to verbs are moved to in front of the verb.

We performed these reorderings on top of the data obtained with the *Noun+Adj+Verb+Adv+PostP* transformations earlier in Section 3.2.2 and used the same decoder parameters. Table 4 shows the performance obtained after various combination of reordering operations over the 10 data sets. Although there were some improvements for certain cases, none of reordering gave consistent improvements for all the data sets. A cursory examinations of the alignments produced after these reordering transformations indicated that the resulting root alignments were not necessarily that close to being monotonic as we would have expected.

Experiment	Ave.	STD	Max.	Min.
Baseline	**21.96**	**0.72**	**22.91**	**20.67**
ObjR	21.94	0.71	23.12	20.56
ObjR+AdvR	21.73	0.50	22.44	20.69
ObjR+PassAgR	21.88	0.73	23.03	20.51
ObjR+SubCR	21.88	0.61	22.77	20.92

Table 4: BLEU scores of after reordering transformations

5 Related Work

Statistical Machine Translation into a morphologically rich language is a challenging problem in that, on the target side, the decoder needs to generate both the right sequence of constituents and the right sequence of morphemes for each word. Furthermore, since for such languages one can generate tens of hundreds of inflected variants, standard word-based alignment approaches suffer from sparseness issues. Koehn (2005) applied standard phrase-based SMT to Finnish using the Europarl corpus and reported that translation to Finnish had the worst BLEU scores.

Using morphology in statistical machine translation has been addressed by many researchers for translation from or into morphologically rich(er) languages. Niessen and Ney (2004) used morphological decomposition to get better alignments. Yang and Kirchhoff (2006) have used phrase-based backoff models to translate unknown words by morphologically decomposing the unknown source words. Lee (2004) and Zolmann et *al.* (2006) have exploited morphology in Arabic-English SMT. Popovic and Ney (2004) investigated improving translation quality from inflected languages by using stems, suffixes and part-of-speech tags. Goldwater and McClosky (2005) use morphological analysis on the Czech side to get improvements in Czech-to-English statistical machine translation. Minkov et *al.* (2007) have used morphological postprocessing on the target side, to improve translation quality. Avramidis and Koehn (2008) have annotated English with additional morphological information extracted from a syntactic tree, and have used this in translation to Greek and Czech. Recently, Bisazza and Federico (2009) have applied morphological segmentation in *Turkish-to-English* statistical machine translation and found that it provides nontrivial BLEU

[16] These experiments were done on top of the model in 3.2.3 with a 3-gram word and root LMs and 8-gram tag LM.

[17] Although Turkish is a free-constituent order language, SOV is the dominant order in text.

score improvements.

In the context of translation from English to Turkish, Durgar-El Kahlout and Oflazer (2010) have explored different representational units of the lexical morphemes and found that selectively splitting morphemes on the target side provided nontrivial improvement in the BLEU score. Their approach was based on splitting the target Turkish side, into constituent morphemes while our approach in this paper is the polar opposite: we do not segment morphemes on the Turkish side but rather join function words on the English side to the related content words. Our approach is somewhat similar to recent approaches that use complex syntactically-motivated complex tags. Birch et *al.* (2007) have integrated more syntax in a factored translation approach by using CCG supertags as a separate factor and have reported a 0.46 BLEU point improvement in Dutch-to-English translations. Although they used supertags, these were obtained not via syntactic analysis but by supertagging, while we determine, on the fly, the appropriate syntactic tags based on syntactic structure. A similar approach based on supertagging was proposed by Hassan et *al.* (2007). They used both CCG supertags and LTAG supertags in Arabic-to-English phrase-based translation and have reported about 6% relative improvement in BLEU scores. In the context of reordering, one recent work (Xu et al., 2009), was able to get an improvement of 0.6 BLEU points by using source syntactic analysis and a constituent reordering scheme like ours for English-to-Turkish translation, but without using any morphology.

6 Conclusions

We have presented a novel way to incorporate source syntactic structure in English-to-Turkish phrase-based machine translation by parsing the source sentences and then encoding many local and nonlocal source syntactic structures as additional complex tag factors. Our goal was to obtain representations of source syntactic structures that parallel target morphological structures, and enable us to extend factored translation, in applicability, to languages with very disparate morphological structures.

In our experiments over a limited amount training data, but repeated with 10 different training and test sets, we found that syntax-to-morphology mapping transformations on the source side sentences, along with a very small set of transformations on the target side, coupled with some additional techniques provided about 39% relative improvement in BLEU scores over a word-based baseline and about 28% improvement of a factored baseline. We also experimented with numerous additional syntactic reordering transformation on the source to further bring the constituent order in line with the target order but found that these did not provide any tangible improvements when averaged over the 10 different data sets.

It is possible that the techniques presented in this paper may be less effective if the available data is much larger, but we have reasons to believe that they will still be effective then also. The reduction in size of the source language side of the training corpus seems to be definitely effective and there no reason why such a reduction (if not more) will not be observed in larger data. Also, the preprocessing of English prepositional phrases and many adverbial phrases usually involve rather long distance relations in the source side syntactic structure[18] and when such structures are coded as complex tags on the nominal or verbal heads, such long distance syntax is effectively "localized" and thus can be better captured with the limited window size used for phrase extraction.

One limitation of the approach presented here is that it is not directly applicable in the reverse direction. The data encoding and set-up can directly be employed to generate English "translation" expressed as a sequence of root and complex tag combinations, but then some of the complex tags could encode various syntactic constructs. To finalize the translation after the decoding step, the function words/tags in the complex tag would then have to be unattached and their proper positions in the sentence would have to be located. The problem is essentially one of generating multiple candidate sentences with the unattached function words ambiguously positioned (say in a lattice) and then use a second language model to rerank these sentences to select the target sentence. This is an avenue of research that we intend to look at in the very near future.

Acknowledgements

We thank Joakim Nivre for providing us with the parser. This publication was made possible by the generous support of the Qatar Foundation through Carnegie Mellon University's Seed Research program. The statements made herein are solely the responsibility of the authors.

[18] For instance, consider the example in Figure 2 involving *if* with some additional modifiers added to the intervening noun phrase.

References

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceedings of ACL-08/HLT*, pages 763–770, Columbus, Ohio, June.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored translation models. In *Proceedings of SMT Workshop at the 45th ACL*.

Arianna Bisazza and Marcello Federico. 2009. Morphological pre-processing for Turkish to English statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, Tokyo, Japan, December.

İlknur Durgar-El-Kahlout and Kemal Oflazer. 2010. Exploiting morphology and local word reordering in English to Turkish phrase-based statistical machine translation. *IEEE Transactions on Audio, Speech, and Language Processing*. To Appear.

Alexander Fraser. 2009. Experiments in morphosyntactic processing for translating to and from German. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 115–119, Athens, Greece, March. Association for Computational Linguistics.

Sharon Goldwater and David McClosky. 2005. Improving statistical MT through morphological analysis. In *Proceedings of HLT/EMNLP-2005*, pages 676–683, Vancouver, British Columbia, Canada, October.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th ACL*, pages 288–295, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of EMNLP*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL-2003*.

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

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

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of HLT/NAACL-2004 – Companion Volume*, pages 57–60.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of the 45th ACL*, pages 128–135, Prague, Czech Republic, June. Association for Computational Linguistics.

Sonja Niessen and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntatic information. *Computational Linguistics*, 30(2):181–204.

Joakim Nivre, Hall Johan, Nilsson Jens, Chanev Atanas, Gülşen Eryiğit, Sandra Kübler, Marinov Stetoslav, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering Journal*, 13(2):99–135.

Kemal Oflazer and İlknur Durgar-El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of Statistical Machine Translation Workshop at the 45th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Kemal Oflazer. 2008. Statistical machine translation into a morphologically complex language. In *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 376–387.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318.

Maja Popovic and Hermann Ney. 2004. Towards the use of word stems and suffixes for statistical machine translation. In *Proceedings of the 4th LREC*, pages 1585–1588, May.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT/NAACL-2003*, pages 252–259.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings HLT/NAACL-2009*, pages 245–253, June.

Mei Yang and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of EACL-2006*, pages 41–48.

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for Turkish. In *Proceedings of HLT/NAACL-2006*, pages 328–334, New York City, USA, June.

Andreas Zollmann, Ashish Venugopal, and Stephan Vogel. 2006. Bridging the inflection morphology gap for Arabic statistical machine translation. In *Proceedings of HLT/NAACL-2006 – Companion Volume*, pages 201–204, New York City, USA, June.

Hindi-to-Urdu Machine Translation Through Transliteration

Nadir Durrani **Hassan Sajjad** **Alexander Fraser** **Helmut Schmid**
Institute for Natural Language Processing
University of Stuttgart
{durrani,sajjad,fraser,schmid}@ims.uni-stuttgart.de

Abstract

We present a novel approach to integrate transliteration into Hindi-to-Urdu statistical machine translation. We propose two probabilistic models, based on conditional and joint probability formulations, that are novel solutions to the problem. Our models consider both transliteration and translation when translating a particular Hindi word given the context whereas in previous work transliteration is only used for translating OOV (out-of-vocabulary) words. We use transliteration as a tool for disambiguation of Hindi homonyms which can be both translated or transliterated or transliterated differently based on different contexts. We obtain final BLEU scores of 19.35 (conditional probability model) and 19.00 (joint probability model) as compared to 14.30 for a baseline phrase-based system and 16.25 for a system which transliterates OOV words in the baseline system. This indicates that transliteration is useful for more than only translating OOV words for language pairs like Hindi-Urdu.

1 Introduction

Hindi is an official language of India and is written in Devanagari script. Urdu is the national language of Pakistan, and also one of the state languages in India, and is written in Perso-Arabic script. Hindi inherits its vocabulary from Sanskrit while Urdu descends from several languages including Arabic, Farsi (Persian), Turkish and Sanskrit. Hindi and Urdu share grammatical structure and a large proportion of vocabulary that they both inherited from Sanskrit. Most of the verbs and closed-class words (pronouns, auxiliaries, case-markers, etc) are the same. Because both languages have lived together for centuries, some Urdu words which originally came from Arabic and Farsi have also mixed into Hindi and are now part of the Hindi vocabulary. The spoken form of the two languages is very similar.

The extent of overlap between Hindi and Urdu vocabulary depends upon the domain of the text. Text coming from the literary domain like novels or history tend to have more Sanskrit (for Hindi) and Persian/Arabic (for Urdu) vocabulary. However, news wire that contains text related to media, sports and politics, etc., is more likely to have common vocabulary.

In an initial study on a small news corpus of 5000 words, randomly selected from BBC[1] News, we found that approximately 62% of the Hindi types are also part of Urdu vocabulary and thus can be transliterated while only 38% have to be translated. This provides a strong motivation to implement an end-to-end translation system which strongly relies on high quality transliteration from Hindi to Urdu.

Hindi and Urdu have similar sound systems but transliteration from Hindi to Urdu is still very hard because some phonemes in Hindi have several orthographic equivalents in Urdu. For example the "z" sound[2] can only be written as ज़ whenever it occurs in a Hindi word but can be written as ذ, ز, ض and ظ in an Urdu word. Transliteration becomes non-trivial in cases where the multiple orthographic equivalents for a Hindi word are all valid Urdu words. Context is required to resolve ambiguity in such cases. Our transliterator (described in sections 3.1.2 and 4.1.3) gives an accuracy of 81.6% and a 25-best accuracy of 92.3%.

Transliteration has been previously used only as a back-off measure to translate NEs (Name Entities) and OOV words in a pre- or post-processing step. The problem we are solving is more difficult than techniques aimed at handling OOV words,

[1] http://www.bbc.co.uk/hindi/index.shtml
[2] All sounds are represented using SAMPA notation.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 465–474,
Uppsala, Sweden, 11-16 July 2010.

Hindi	Urdu	SAMPA	Gloss
आम	عام/آم	Am	Mango/Ordinary
जाली	جالی/جعلی	d_ZAli	Fake/Net
शेर	شعر/شیر	Ser	Lion/Verse

Table 1: Hindi Words That Can Be Transliterated Differently in Different Contexts

Hindi	Urdu	SAMPA	Gloss
सीमा	سیما/سرحد	simA	Border/Seema
अंबर	امبر/آسمان	Amb@r	Sky/Ambar
विजय	وِجے/جیت	vId_Ze	Victory/Vijay

Table 2: Hindi Words That Can Be Translated or Transliterated in Different Contexts

which focus primarily on name transliteration, because we need different transliterations in different contexts; in their case context is irrelevant. For example: consider the problem of transliterating the English word "read" to a phoneme representation in the context "I will read" versus the context "I have read". An example of this for Hindi to Urdu transliteration: the two Urdu words صورت (face/condition) and سورت (chapter of the Koran) are both written as सूरत (sur@t_d) in Hindi. The two are pronounced identically in Urdu but written differently. In such cases we hope to choose the correct transliteration by using context. Some other examples are shown in Table 1.

Sometimes there is also an ambiguity of whether to translate or transliterate a particular word. The Hindi word शान्ती, for example, will be translated to سکون (peace, s@kun) when it is a common noun but transliterated to شانتی (Shanti, SAnt_di) when it is a proper name. We try to model whether to translate or transliterate in a given situation. Some other examples are shown in Table 2.

The remainder of this paper is organized as follows. Section 2 provides a review of previous work. Section 3 introduces two probabilistic models for integrating translations and transliterations into a translation model which are based on conditional and joint probability distributions. Section 4 discusses the training data, parameter optimization and the initial set of experiments that compare our two models with a baseline Hindi-Urdu phrase-based system and with two transliteration-aided phrase-based systems in terms of BLEU scores (Papineni et al., 2001). Section 5 performs an error analysis showing interesting weaknesses in the initial formulations. We remedy the problems by adding some heuristics and modifications to our models which show improvements in the results as discussed in section 6. Section 7 gives two examples illustrating how our model decides whether to translate or transliterate and how it is able to choose among different valid transliterations given the context. Section 8 concludes the paper.

2 Previous Work

There has been a significant amount of work on transliteration. We can break down previous work into three groups. The first group is generic transliteration work, which is evaluated outside of the context of translation. This work uses either grapheme or phoneme based models to transliterate words lists (Knight and Graehl, 1998; Li et al., 2004; Ekbal et al., 2006; Malik et al., 2008). The work by Malik et al. addresses Hindi to Urdu transliteration using hand-crafted rules and a phonemic representation; it ignores translation context.

A second group deals with out-of-vocabulary words for SMT systems built on large parallel corpora, and therefore focuses on name transliteration, which is largely independent of context. Al-Onaizan and Knight (2002) transliterate Arabic NEs into English and score them against their respective translations using a modified IBM Model 1. The options are further re-ranked based on different measures such as web counts and using co-reference to resolve ambiguity. These re-ranking methodologies can not be performed in SMT at the decoding time. An efficient way to compute and re-rank the transliterations of NEs and integrate them on the fly might be possible. However, this is not practical in our case as our model considers transliterations of all input words and not just NEs. A log-linear block transliteration model is applied to OOV NEs in Arabic to English SMT by Zhao et al. (2007). This work is also transliterating only NEs and not doing any disambiguation. The best method proposed by Kashani et al. (2007) integrates translations provided by external sources such as transliteration or rule-base translation of numbers and dates, for an arbitrary number of entries within the input text. Our work is different from Kashani et al. (2007) in that our model compares transliterations with translations

on the fly whereas transliterations in Kashani et al. do not compete with internal phrase tables. They only compete amongst themselves during a second pass of decoding. Hermjakob et al. (2008) use a tagger to identify good candidates for transliteration (which are mostly NEs) in input text and add transliterations to the SMT phrase table dynamically such that they can directly compete with translations during decoding. This is closer to our approach except that we use transliteration as an alternative to translation for all Hindi words. Our focus is disambiguation of Hindi homonyms whereas they are concentrating only on transliterating NE's. Moreover, they are working with a large bitext so they can rely on their translation model and only need to transliterate NEs and OOVs. Our translation model is based on data which is both sparse and noisy. Therefore we pit transliterations against translations for every input word. Sinha (2009) presents a rule-based MT system that uses Hindi as a pivot to translate from English to Urdu. This work also uses transliteration only for the translation of unknown words. Their work can not be used for direct translation from Hindi to Urdu (independently of English) "due to various ambiguous mappings that have to be resolved".

The third group uses transliteration models inside of a cross-lingual IR system (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003; Pirkola et al., 2003). Picking a single best transliteration or translation in context is not important in an IR system. Instead, all the options are used by giving them weights and context is typically not taken into account.

3 Our Approach

Both of our models combine a character-based transliteration model with a word-based translation model. Our models look for the most probable Urdu token sequence u_1^n for a given Hindi token sequence h_1^n. We assume that each Hindi token is mapped to exactly one Urdu token and that there is no reordering. The assumption of no reordering is reasonable given the fact that Hindi and Urdu have identical grammar structure and the same word order. An Urdu token might consist of more than one Urdu word[3]. The following sections give a mathematical formulation of our two models, Model-1 and Model-2.

3.1 Model-1 : Conditional Probability Model

Applying a noisy channel model to compute the most probable translation $\hat{u}_1^n$, we get:

$$\arg\max_{u_1^n} p(u_1^n|h_1^n) = \arg\max_{u_1^n} p(u_1^n)p(h_1^n|u_1^n) \quad (1)$$

3.1.1 Language Model

The **language model (LM)** $p(u_1^n)$ is implemented as an n-gram model using the SRILM-Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. The parameters of the language model are learned from a monolingual Urdu corpus. The language model is defined as:

$$p(u_1^n) = \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1}) \quad (2)$$

where k is a parameter indicating the amount of context used (e.g., $k = 4$ means 5-gram model). u_i can be a single or a multi-word token. A multi-word token consists of two or more Urdu words. For a multi-word u_i we do multiple language model look-ups, one for each u_{i_x} in $u_i = u_{i_1}, \ldots, u_{i_m}$ and take their product to obtain the value $p_{LM}(u_i|u_{i-k}^{i-1})$.

Language Model for Unknown Words: Our model generates transliterations that can be known or unknown to the language model and the translation model. We refer to the words known to the language model and to the translation model as LM-known and TM-known words respectively and to words that are unknown as LM-unknown and TM-unknown respectively.

We assign a special value ψ to the LM-unknown words. If one or more u_{i_x} in a multi-word u_i are LM-unknown we assign a language model score $p_{LM}(u_i|u_{i-k}^{i-1}) = \psi$ for the entire u_i, meaning that we consider partially known transliterations to be as bad as fully unknown transliterations. The parameter ψ controls the trade-off between LM-known and LM-unknown transliterations. It does not influence translation options because they are always LM-known in our case. This is because our monolingual corpus also contains the Urdu part of translation corpus. The optimization of ψ is described in section 4.2.1.

[3]This occurs frequently in case markers with nouns, derivational affixes and compounds etc. These are written as single words in Hindi as opposed to Urdu where they are written as two words. For example खूबसूरत (beautiful ; xubsur@t_d) and आपका (your's ; ApkA) are written as خوب صورت and آپ کا respectively in Urdu.

3.1.2 Translation Model

The **translation model (TM)** $p(h_1^n|u_1^n)$ is approximated with a context-independent model:

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} p(h_i|u_i) \quad (3)$$

where h_i and u_i are Hindi and Urdu tokens respectively. Our model estimates the conditional probability $p(h_i|u_i)$ by interpolating a word-based model and a character-based (transliteration) model.

$$p(h_i|u_i) = \lambda p_w(h_i|u_i) + (1-\lambda)p_c(h_i|u_i) \quad (4)$$

The parameters of the **word-based translation model** $p_w(h|u)$ are estimated from the word alignments of a small parallel corpus. We only retain 1-1/1-N (1 Hindi word, 1 or more Urdu words) alignments and throw away N-1 and M-N alignments for our models. This is further discussed in section 4.1.1.

The **character-based transliteration model** $p_c(h|u)$ is computed in terms of $p_c(h,u)$, a joint character model, which is also used for Chinese-English back-transliteration (Li et al., 2004) and Bengali-English name transliteration (Ekbal et al., 2006). The character-based transliteration probability is defined as follows:

$$\begin{aligned} p_c(h,u) &= \sum_{a_1^n \in align(h,u)} p(a_1^n) \\ &= \sum_{a_1^n \in align(h,u)} \prod_{i=1}^{n} p(a_i|a_{i-k}^{i-1}) \end{aligned} \quad (5)$$

where a_i is a pair consisting of the i-th Hindi character h_i and the sequence of 0 or more Urdu characters that it is aligned with. A sample alignment is shown in Table 3(b) in section 4.1.3. Our best results are obtained with a 5-gram model. The parameters $p(a_i|a_{i-k}^{i-1})$ are estimated from a small transliteration corpus which we automatically extracted from the translation corpus. The extraction details are also discussed in section 4.1.3. Because our overall model is a conditional probability model, joint-probabilities are marginalized using character-based prior probabilities:

$$p_c(h|u) = \frac{p_c(h,u)}{p_c(u)} \quad (6)$$

The **prior probability** $p_c(u)$ of the character sequence $u = c_1^m$ is defined with a character-based language model:

$$p_c(u) = \prod_{i=1}^{m} p(c_i|c_{i-k}^{i-1}) \quad (7)$$

The parameters $p(c_i|c_{i-k}^{i-1})$ are estimated from the Urdu part of the character-aligned transliteration corpus. Replacing (6) in (4) we get:

$$p(h_i|u_i) = \lambda p_w(h_i|u_i) + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)} \quad (8)$$

Having all the components of our model defined we insert (8) and (2) in (1) to obtain the final equation:

$$\begin{aligned} \hat{u}_1^n = \arg\max_{u_1^n} \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1})[\lambda p_w(h_i|u_i) \\ + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}] \end{aligned} \quad (9)$$

The optimization of the interpolating factor λ is discussed in section 4.2.1.

3.2 Model-2 : Joint Probability Model

This section briefly defines a variant of our model where we interpolate joint probabilities instead of conditional probabilities. Again, the **translation model** $p(h_1^n|u_1^n)$ is approximated with a context-independent model:

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} p(h_i|u_i) = \prod_{i=1}^{n} \frac{p(h_i,u_i)}{p(u_i)} \quad (10)$$

The joint probability $p(h_i,u_i)$ of a Hindi and an Urdu word is estimated by interpolating a word-based model and a character-based model.

$$p(h_i,u_i) = \lambda p_w(h_i,u_i) + (1-\lambda)p_c(h_i,u_i) \quad (11)$$

and the prior probability $p(u_i)$ is estimated as:

$$p(u_i) = \lambda p_w(u_i) + (1-\lambda)p_c(u_i) \quad (12)$$

The parameters of the translation model $p_w(h_i,u_i)$ and the word-based prior probabilities $p_w(u_i)$ are estimated from the 1-1/1-N word-aligned corpus (the one that we also used to estimate translation probabilities $p_w(h_i|u_i)$ previously).

The character-based transliteration probability $p_c(h_i,u_i)$ and the character-based prior probability $p_c(u_i)$ are defined by (5) and (7) respectively in

the previous section. Putting (11) and (12) in (10) we get

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} \frac{\lambda p_w(h_i, u_i) + (1-\lambda) p_c(h_i, u_i)}{\lambda p_w(u_i) + (1-\lambda) p_c(u_i)} \quad (13)$$

The idea is to interpolate joint probabilities and divide them by the interpolated marginals. The final equation for Model-2 is given as:

$$\hat{u}_1^n = \arg\max_{u_1^n} \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1}) \times \frac{\lambda p_w(h_i, u_i) + (1-\lambda) p_c(h_i, u_i)}{\lambda p_w(u_i) + (1-\lambda) p_c(u_i)} \quad (14)$$

3.3 Search

The decoder performs a stack-based search using a beam-search algorithm similar to the one used in Pharoah (Koehn, 2004a). It searches for an Urdu string that maximizes the product of translation probability and the language model probability (equation 1) by translating one Hindi word at a time. It is implemented as a two-level process. At the lower level, it computes n-best transliterations for each Hindi word h_i according to $p_c(h, u)$. The joint probabilities given by $p_c(h, u)$ are marginalized for each Urdu transliteration to give $p_c(h|u)$. At the higher level, transliteration probabilities are interpolated with $p_w(h|u)$ and then multiplied with language model probabilities to give the probability of a hypothesis. We use 20-best translations and 25-best transliterations for $p_w(h|u)$ and $p_c(h|u)$ respectively and a 5-gram language model.

To keep the search space manageable and time complexity polynomial we apply pruning and recombination. Since our model uses monotonic decoding we only need to recombine hypotheses that have the same context (last n-1 words). Next we do histogram-based pruning, maintaining the 100-best hypotheses for each stack.

4 Evaluation

4.1 Training

This section discusses the training of the different model components.

4.1.1 Translation Corpus

We used the freely available EMILLE Corpus as our bilingual resource which contains roughly 13,000 Urdu and 12,300 Hindi sentences. From these we were able to sentence-align 7000 sentence pairs using the sentence alignment algorithm given by Moore (2002).

The word alignments for this task were extracted by using GIZA++ (Och and Ney, 2003) in both directions. We extracted a total of 107323 alignment pairs (5743 N-1 alignments, 8404 M-N alignments and 93176 1-1/1-N alignments). Of these alignments M-N and N-1 alignment pairs were ignored. We manually inspected a sample of 1000 instances of M-N/N-1 alignments and found that more than 70% of these were (totally or partially) wrong. Of the 30% correct alignments, roughly one-third constitute N-1 alignments. Most of these are cases where the Urdu part of the alignment actually consists of two (or three) words but was written without space because of lack of standard writing convention in Urdu. For example جا سکتے (can go ; d_ZA s@kt_de) is alternatively written as جاسکتے (can go ; d_ZAs@kt_de) i.e. without space. We learned that these N-1 translations could be safely dropped because we can generate a separate Urdu word for each Hindi word. For valid M-N alignments we observed that these could be broken into 1-1/1-N alignments in most of the cases. We also observed that we usually have coverage of the resulting 1-1 and 1-N alignments in our translation corpus. Looking at the noise in the incorrect alignments we decided to drop N-1 and M-N cases. We do not model deletions and insertions so we ignored null alignments. Also 1-N alignments with gaps were ignored. Only the alignments with contiguous words were kept.

4.1.2 Monolingual Corpus

Our monolingual Urdu corpus consists of roughly 114K sentences. This comprises 108K sentences from the data made available by the University of Leipzig[4] + 5600 sentences from the training data of each fold during cross validation.

4.1.3 Transliteration Corpus

The training corpus for transliteration is extracted from the 1-1/1-N word-alignments of the EMILLE corpus discussed in section 4.1.1. We use an edit distance algorithm to align this training corpus at the character level and we eliminate translation pairs with high edit distance which are unlikely to be transliterations.

[4] http://corpora.informatik.uni-leipzig.de/

We used our knowledge of the Hindi and Urdu scripts to define the initial character mapping. The mapping was further extended by looking into available Hindi-Urdu transliteration systems[5,6] and other resources (Gupta, 2004; Malik et al., 2008; Jawaid and Ahmed, 2009). Each pair in the character map is assigned a cost. A Hindi character that always map to only one Urdu character is assigned a cost of 0 whereas the Hindi characters that map to different Urdu characters are assigned a cost of 0.2. The edit distance metric allows insert, delete and replace operations. The handcrafted pairs define the cost of replace operations. We set a cost of 0.6 for deletions and insertions. These costs were optimized on held out data. The details of optimization are not mentioned due to limited space. Using this metric we filter out the word pairs with high edit-distance to extract our transliteration corpus. We were able to extract roughly 2100 unique pairs along with their alignments. The resulting alignments are modified by merging unaligned $\emptyset \rightarrow 1$ (no character on source side, 1 character on target side) or $\emptyset \rightarrow N$ alignments with the preceding alignment pair. If there is no preceding alignment pair then it is merged with the following pair. Table 3 gives an example showing initial alignment (a) and the final alignment (b) after applying the merge operation. Our model retains $1 \rightarrow \emptyset$ and $N \rightarrow \emptyset$ alignments as deletion operations.

a)	Hindi	$\emptyset$	b	c	$\emptyset$	e	f
	Urdu	A	XY	C	D	$\emptyset$	F
b)	Hindi		b	c		e	f
	Urdu		AXY	CD		$\emptyset$	F

Table 3: Alignment (a) Before (b) After Merge

The parameters $p_c(h, u)$ and $p_c(u)$ are trained on the aligned corpus using the SRILM toolkit. We use Add-1 smoothing for unigrams and Kneser-Ney smoothing for higher n-grams.

4.1.4 Diacritic Removal and Normalization

In Urdu, short vowels are represented with diacritics but these are rarely written in practice. In order to keep the data consistent, all diacritics are removed. This loss of information is not harmful when transliterating/translating from Hindi to Urdu because undiacritized text is equally readable to native speakers as its diacritized counter part. However leaving occasional diacritics in the corpus can worsen the problem of data sparsity by creating spurious ambiguity[7].

There are a few Urdu characters that have multiple equivalent Unicodes. All such forms are normalized to have only one representation[8].

4.2 Experimental Setup

We perform a 5-fold cross validation taking 4/5 of the data as training and 1/5 as test data. Each fold comprises roughly 1400 test sentences and 5600 training sentences.

4.2.1 Parameter Optimization

Our model contains two parameters λ (the interpolating factor between translation and transliteration modules) and ψ (the factor that controls the trade-off between LM-known and LM-unknown transliterations). The interpolating factor λ is initialized, inspired by Written-Bell smoothing, with a value of $\frac{N}{N+B}$[9]. We chose a very low value $1e^{-40}$ for the factor ψ initially, favoring LM-known transliterations very strongly. Both of these parameters are optimized as described below.

Because our training data is very sparse we do not use held-out data for parameter optimization. Instead we optimize these parameters by performing a 2-fold optimization for each of the 5 folds. Each fold is divided into two halves. The parameters λ and ψ are optimized on the first half and the other half is used for testing, then optimization is done on the second half and the first half is used for testing. The optimal value for parameter λ occurs between 0.7-0.84 and for the parameter ψ between $1e^{-5}$ and $1e^{-10}$.

4.2.2 Results

Baseline Pb_0: We ran Moses (Koehn et al., 2007) using Koehn's training scripts[10], doing a 5-fold cross validation with no reordering[11]. For the other parameters we use the default values i.e. 5-gram language model and maximum phrase-length= 6. Again, the language model is imple-

[5]CRULP: http://www.crulp.org/software/langproc.htm

[6]Malerkotla.org: http://translate.malerkotla.co.in

[7]It should be noted though that diacritics play a very important role when transliterating in the reverse direction because these are virtually always written in Hindi as dependent vowels.

[8]www.crulp.org/software/langproc/urdunormalization.htm

[9]N is the number of aligned word pairs (tokens) and B is the number of different aligned word pairs (types).

[10]http://statmt.org/wmt08/baseline.html

[11]Results are worse with reordering enabled.

M	Pb_0	Pb_1	Pb_2	M_1	M_2
BLEU	14.3	16.25	16.13	18.6	17.05

Table 4: Comparing Model-1 and Model-2 with Phrase-based Systems

mented as an n-gram model using the SRILM-Toolkit with Kneser-Ney smoothing. Each fold comprises roughly 1400 test sentences, 5000 in training and 600 in dev[12]. We also used two methods to incorporate transliterations in the phrase-based system:

Post-process Pb_1: All the OOV words in the phrase-based output are replaced with their top-candidate transliteration as given by our transliteration system.

Pre-process Pb_2: Instead of adding transliterations as a post process we do a second pass by adding the unknown words with their top-candidate transliteration to the training corpus and rerun Koehn's training script with the new training corpus. Table 4 shows results (taking arithmetic average over 5 folds) from Model-1 and Model-2 in comparison with three baselines discussed above.

Both our systems (Model-1 and Model-2) beat the baseline phrase-based system with a BLEU point difference of 4.30 and 2.75 respectively. The transliteration aided phrase-based systems Pb_1 and Pb_2 are closer to our Model-2 results but are way below Model-1 results. The difference of 2.35 BLEU points between M_1 and Pb_1 indicates that transliteration is useful for more than only translating OOV words for language pairs like Hindi-Urdu. Our models choose between translations and transliterations based on context unlike the phrase-based systems Pb_1 and Pb_2 which use transliteration only as a tool to translate OOV words.

5 Error Analysis

Based on preliminary experiments we found three major flaws in our initial formulations. This section discusses each one of them and provides some heuristics and modifications that we employ to try to correct deficiencies we found in the two models described in section 3.1 and 3.2.

[12]After having the MERT parameters, we add the 600 dev sentences back into the training corpus, retrain GIZA, and then estimate a new phrase table on all 5600 sentences. We then use the MERT parameters obtained before together with the newer (larger) phrase-table set.

5.1 Heuristic-1

A lot of errors occur because our translation model is built on very sparse and noisy data. The motivation for this heuristic is to counter wrong alignments at least in the case of verbs and functional words (which are often transliterations). This heuristic favors translations that also appear in the n-best transliteration list over only-translation and only-transliteration options. We modify the translation model for both the conditional and the joint model by adding another factor which strongly weighs translation+transliteration options by taking the square-root of the product of the translation and transliteration probabilities. Thus modifying equations (8) and (11) in Model-1 and Model-2 we obtain equations (15) and (16) respectively:

$$p(h_i|u_i) = \lambda_1 p_w(h_i|u_i) + \lambda_2 \frac{p_c(h_i,u_i)}{p_c(u_i)} + \lambda_3 \sqrt{p_w(h_i|u_i)\frac{p_c(h_i,u_i)}{p_c(u_i)}} \quad (15)$$

$$p(h_i,u_i) = \lambda_1 p_w(h_i,u_i) + \lambda_2 p_c(h_i,u_i) + \lambda_3 \sqrt{p_w(h_i,u_i)p_c(h_i,u_i)} \quad (16)$$

For the optimization of lambda parameters we hold the value of the translation coefficient λ_1[13] and the transliteration coefficient λ_2 constant (using the optimized values as discussed in section 4.2.1) and optimize λ_3 again using 2-fold optimization on all the folds as described above[14].

5.2 Heuristic-2

When an unknown Hindi word occurs for which all transliteration options are LM-unknown then the best transliteration should be selected. The problem in our original models is that a fixed LM probability ψ is used for LM-unknown transliterations. Hence our model selects the transliteration that has the best $\frac{p_c(h_i,u_i)}{p_c(u_i)}$ score i.e. we maximize $p_c(h_i|u_i)$ instead of $p_c(u_i|h_i)$ (or equivalently $p_c(h_i,u_i)$). The reason is an inconsistency in our models. The language model probability of unknown words is uniform (and equal to ψ) whereas the translation model uses the non-uniform prior probability $p_c(u_i)$ for these words. There is another reason why we can not use the

[13]The translation coefficient λ_1 is same as λ used in previous models and the transliteration coefficient $\lambda_2 = 1 - \lambda$

[14]After optimization we normalize the lambdas to make their sum equal to 1.

value ψ in this case. Our transliterator model also produces space inserted words. The value of ψ is very small because of which transliterations that are actually LM-unknown, but are mistakenly broken into constituents that are LM-known, will always be preferred over their counter parts. An example of this is अमेरिका (America) for which two possible transliterations as given by our model are امیرکا (AmerIkA, without space) and امیر کا (AmerI kA, with space). The latter version is LM-known as its constituents are LM-known. Our models always favor the latter version. Space insertion is an important feature of our transliteration model. We want our transliterator to tackle compound words, derivational affixes, case-markers with nouns that are written as one word in Hindi but as two or more words in Urdu. Examples were already shown in section 3's footnote.

We eliminate the inconsistency by using $p_c(u_i)$ as the 0-gram back-off probability distribution in the language model. For an LM-unknown transliterations we now get in Model-1:

$$
\begin{aligned}
& p(u_i|u_{i-k}^{i-1})[\lambda p_w(h_i|u_i) + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}] \\
&= p(u_i|u_{i-k}^{i-1})[(1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}] \\
&= \prod_{j=0}^{k} \alpha(u_{i-j}^{i-1}) p_c(u_i)[(1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}] \\
&= \prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})[(1-\lambda) p_c(h_i,u_i)]
\end{aligned}
$$

where $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})$ is just the constant that SRILM returns for unknown words. The last line of the calculation shows that we simply drop $p_c(u_i)$ if u_i is LM-unknown and use the constant $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})$ instead of ψ. A similar calculation for Model-2 gives $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1}) p_c(h_i,u_i)$.

5.3 Heuristic-3

This heuristic discusses a flaw in Model-2. For transliteration options that are TM-unknown, the $p_w(h,u)$ and $p_w(u)$ factors becomes zero and the translation model probability as given by equation (13) becomes:

$$\frac{(1-\lambda)p_c(h_i,u_i)}{(1-\lambda)p_c(u_i)} = \frac{p_c(h_i,u_i)}{p_c(u_i)}$$

In such cases the λ factor cancels out and no weighting of word translation vs. transliteration occurs anymore. As a result of this, transliterations are sometimes incorrectly favored over their translation alternatives.

In order to remedy this problem we assign a minimal probability β to the word-based prior $p_w(u_i)$ in case of TM-unknown transliterations, which prevents it from ever being zero. Because of this addition the translation model probability for LM-unknown words becomes:

$$\frac{(1-\lambda)p_c(h_i,u_i)}{\lambda\beta + (1-\lambda)p_c(u_i)} \text{ where } \beta = \frac{1}{\text{Urdu Types in TM}}$$

	$\mathbf{H_1}$	$\mathbf{H_2}$	$\mathbf{H_{12}}$
$\mathbf{M_1}$	18.86	18.97	19.35
$\mathbf{M_2}$	17.56	17.85	18.34

Table 5: Applying Heuristics 1 and 2 and their Combinations to Model-1 and Model-2

	$\mathbf{H_3}$	$\mathbf{H_{13}}$	$\mathbf{H_{23}}$	$\mathbf{H_{123}}$
$\mathbf{M_2}$	18.52	18.93	18.55	19.00

Table 6: Applying Heuristic 3 and its Combinations with other Heuristics to Model-2

6 Final Results

This section shows the improvement in BLEU score by applying heuristics and combinations of heuristics in both the models. Tables 5 and 6 show the improvements achieved by using the different heuristics and modifications discussed in section 5. We refer to the results as $M_x H_y$ where x denotes the model number, 1 for the conditional probability model and 2 for the joint probability model and y denotes a heuristic or a combination of heuristics applied to that model[15].

Both heuristics (H_1 and H_2) show improvements over their base models M_1 and M_2. Heuristic-1 shows notable improvement for both models in parts of test data which has high number of common vocabulary words. Using heuristic 2 we were able to properly score LM-unknown transliterations against each other. Using these heuristics together we obtain a gain of 0.75 over M-1 and a gain of 1.29 over M-2.

Heuristic-3 remedies the flaw in M_2 by assigning a special value to the word-based prior $p_w(u_i)$ for TM-unknown words which prevents the cancelation of interpolating parameter λ. M_2 combined with heuristic 3 (M_2H_3) results in a 1.47

[15]For example M_1H_1 refers to the results when heuristic-1 is applied to model-1 whereas M_2H_{12} refers to the results when heuristics 1 and 2 are together applied to model 2.

BLEU point improvement and combined with all the heuristics (M_2H_{123}) gives an overall gain of 1.95 BLEU points and is close to our best results (M_1H_{12}). We also performed significance test by concatenating all the fold results. Both our best systems M_1H_{12} and M_2H_{123} are statistically significant ($p < 0.05$)[16] over all the baselines discussed in section 4.2.2.

One important issue that has not been investigated yet is that BLEU has not yet been shown to have good performance in morphologically rich target languages like Urdu, but there is no metric known to work better. We observed that sometimes on data where the translators preferred to translate rather than doing transliteration our system is penalized by BLEU even though our output string is a valid translation. For other parts of the data where the translators have heavily used transliteration, the system may receive a higher BLEU score. We feel that this is an interesting area of research for automatic metric developers, and that a large scale task of translation to Urdu which would involve a human evaluation campaign would be very interesting.

7 Sample Output

This section gives two examples showing how our model ($M1H2$) performs disambiguation. Given below are some test sentences that have Hindi homonyms (underlined in the examples) along with Urdu output given by our system. In the first example (given in Figure 1) Hindi word शेर can be transliterated to شیر (Lion) or شعر (Verse) depending upon the context. Our model correctly identifies which transliteration to choose given the context.

In the second example (shown in Figure 2) Hindi word शान्ती can be translated to سکون (peace, s@kun) when it is a common noun but transliterated to شانتی (Shanti, SAnt_di) when it is a proper name. Our model successfully decides whether to translate or transliterate given the context.

8 Conclusion

We have presented a novel way to integrate transliterations into machine translation. In closely related language pairs such as Hindi-Urdu with a significant amount of vocabulary overlap, transliteration can be very effective in machine translation for more than just translating OOV words. We have addressed two problems. First, transliteration helps overcome the problem of data sparsity and noisy alignments. We are able to generate word translations that are unseen in the translation corpus but known to the language model. Additionally, we can generate novel transliterations (that are LM-Unknown). Second, generating multiple transliterations for homograph Hindi words and using language model context helps us solve the problem of disambiguation. We found that the joint probability model performs almost as well as the conditional probability model but that it was more complex to make it work well.

शेर जंगल का राजा है
شیر جنگل کا راجہ ہے
Ser d_Z@ngl kA rAd_ZA he
"Lion is the king of jungle"
इकबाल का एक खूबसूरत शेर है
اقبال کا ایک خوب صورت شعر ہے
AIqbAl kA Aek xub sur@t_d Ser he
"There is a beautiful verse from Iqbal"

Figure 1: Different Transliterations in Different Contexts

फिर भी वह शान्ती से नहीं रह सकता है
پھر بھی وہ سکون سے نہیں رہ سکتا ہے
p_hIr b_hi vh s@kun se n@heR̃h s@kt_dA
"Even then he can't live peacefully"
ओम शान्ती ओम फराह खान की दूसरी फिल्म है
اوم شانتی اوم فراح خان کی دوسری فلم ہے
Aom SAnt_di Aom frhA xAn ki d_dusri fIl@m he
"Om Shanti Om is Farah Khan's second film"

Figure 2: Translation or Transliteration

Acknowledgments

The first two authors were funded by the Higher Education Commission (HEC) of Pakistan. The third author was funded by Deutsche Forschungsgemeinschaft grants SFB 732 and MorphoSynt. The fourth author was funded by Deutsche Forschungsgemeinschaft grant SFB 732.

[16] We used Kevin Gimpel's tester (http://www.ark.cs.cmu.edu/MT/) which uses bootstrap resampling (Koehn, 2004b), with 1000 samples.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM 03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 139–146.

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 400–408.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL poster sessions*, pages 191–198, Sydney, Australia. Association for Computational Linguistics.

Swati Gupta. 2004. Aligning Hindi and Urdu bilingual corpora for robust projection. Masters project dissertation, Department of Computer Science, University of Sheffield.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio. Association for Computational Linguistics.

Bushra Jawaid and Tafseer Ahmed. 2009. Hindi to Urdu conversion: beyond simple transliteration. In *Conference on Language and Technology 2009*, Lahore, Pakistan.

Mehdi M. Kashani, Eric Joanis, Roland Kuhn, George Foster, and Fred Popowich. 2007. Integration of an Arabic transliteration module into a statistical machine translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 17–24, Prague, Czech Republic. Association for Computational Linguistics.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Demonstration Program*, Prague, Czech Republic.

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Haizhou Li, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 159–166, Barcelona, Spain. Association for Computational Linguistics.

M G Abbas Malik, Christian Boitet, and Pushpak Bhattacharyya. 2008. Hindi Urdu machine transliteration using finite-state transducers. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, UK.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Conference of the Association for Machine Translation in the Americas (AMTA)*.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore A. Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin. 2003. Fuzzy translation of cross-lingual spelling variants. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 345–352, New York, NY, USA. ACM.

R. Mahesh K. Sinha. 2009. Developing English-Urdu machine translation via Hindi. In *Third Workshop on Computational Approaches to Arabic Script-based Languages (CAASL3), MT Summit XII*, Ottawa, Canada.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Bing Zhao, Nguyen Bach, Ian Lane, and Stephan Vogel. 2007. A log-linear block transliteration model based on bi-stream HMMs. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 364–371, Rochester, New York. Association for Computational Linguistics.

Training Phrase Translation Models with Leaving-One-Out

Joern Wuebker and **Arne Mauser** and **Hermann Ney**
Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Germany
<surname>@cs.rwth-aachen.de

Abstract

Several attempts have been made to learn phrase translation probabilities for phrase-based statistical machine translation that go beyond pure counting of phrases in word-aligned training data. Most approaches report problems with over-fitting. We describe a novel leaving-one-out approach to prevent over-fitting that allows us to train phrase models that show improved translation performance on the WMT08 Europarl German-English task. In contrast to most previous work where phrase models were trained separately from other models used in translation, we include all components such as single word lexica and reordering models in training. Using this consistent training of phrase models we are able to achieve improvements of up to 1.4 points in BLEU. As a side effect, the phrase table size is reduced by more than 80%.

1 Introduction

A phrase-based SMT system takes a source sentence and produces a translation by segmenting the sentence into phrases and translating those phrases separately (Koehn et al., 2003). The phrase translation table, which contains the bilingual phrase pairs and the corresponding translation probabilities, is one of the main components of an SMT system. The most common method for obtaining the phrase table is heuristic extraction from automatically word-aligned bilingual training data (Och et al., 1999). In this method, all phrases of the sentence pair that match constraints given by the alignment are extracted. This includes overlapping phrases. At extraction time it does not matter, whether the phrases are extracted from a highly probable phrase alignment or from an unlikely one.

Phrase model probabilities are typically defined as relative frequencies of phrases extracted from word-aligned parallel training data. The joint counts $C(\tilde{f}, \tilde{e})$ of the source phrase $\tilde{f}$ and the target phrase $\tilde{e}$ in the entire training data are normalized by the marginal counts of source and target phrase to obtain a conditional probability

$$p_H(\tilde{f}|\tilde{e}) = \frac{C(\tilde{f}, \tilde{e})}{C(\tilde{e})}. \quad (1)$$

The translation process is implemented as a weighted log-linear combination of several models $h_m(e_1^I, s_1^K, f_1^J)$ including the logarithm of the phrase probability in source-to-target as well as in target-to-source direction. The phrase model is combined with a language model, word lexicon models, word and phrase penalty, and many others. (Och and Ney, 2004) The best translation $\hat{e}_1^{\hat{I}}$ as defined by the models then can be written as

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K, f_1^J) \right\} \quad (2)$$

In this work, we propose to directly train our phrase models by applying a forced alignment procedure where we use the decoder to find a phrase alignment between source and target sentences of the training data and then updating phrase translation probabilities based on this alignment. In contrast to heuristic extraction, the proposed method provides a way of consistently training and using phrase models in translation. We use a modified version of a phrase-based decoder to perform the forced alignment. This way we ensure that all models used in training are identical to the ones used at decoding time. An illustration of the basic

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 475–484,
Uppsala, Sweden, 11-16 July 2010.

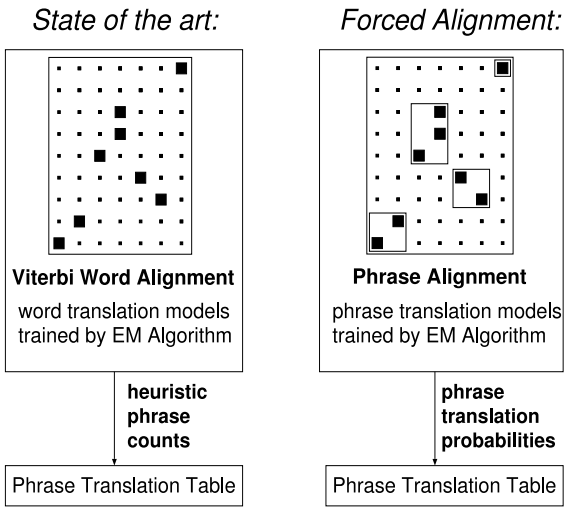


Figure 1: Illustration of phrase training with forced alignment.

idea can be seen in Figure 1. In the literature this method by itself has been shown to be problematic because it suffers from over-fitting (DeNero et al., 2006), (Liang et al., 2006). Since our initial phrases are extracted from the same training data, that we want to align, very long phrases can be found for segmentation. As these long phrases tend to occur in only a few training sentences, the EM algorithm generally overestimates their probability and neglects shorter phrases, which better generalize to unseen data and thus are more useful for translation. In order to counteract these effects, our training procedure applies leaving-one-out on the sentence level. Our results show, that this leads to a better translation quality.

Ideally, we would produce all possible segmentations and alignments during training. However, this has been shown to be infeasible for real-world data (DeNero and Klein, 2008). As training uses a modified version of the translation decoder, it is straightforward to apply pruning as in regular decoding. Additionally, we consider three ways of approximating the full search space:

1. the single-best Viterbi alignment,
2. the n-best alignments,
3. all alignments remaining in the search space after pruning.

The performance of the different approaches is measured and compared on the German-English Europarl task from the *ACL 2008 Workshop on Statistical Machine Translation* (WMT08). Our results show that the proposed phrase model training improves translation quality on the test set by 0.9 BLEU points over our baseline. We find that by interpolation with the heuristically extracted phrases translation performance can reach up to 1.4 BLEU improvement over the baseline on the test set.

After reviewing the related work in the following section, we give a detailed description of phrasal alignment and leaving-one-out in Section 3. Section 4 explains the estimation of phrase models. The empirical evaluation of the different approaches is done in Section 5.

2 Related Work

It has been pointed out in literature, that training phrase models poses some difficulties. For a generative model, (DeNero et al., 2006) gave a detailed analysis of the challenges and arising problems. They introduce a model similar to the one we propose in Section 4.2 and train it with the EM algorithm. Their results show that it can not reach a performance competitive to extracting a phrase table from word alignment by heuristics (Och et al., 1999).

Several reasons are revealed in (DeNero et al., 2006). When given a bilingual sentence pair, we can usually assume there are a number of equally correct phrase segmentations and corresponding alignments. For example, it may be possible to transform one valid segmentation into another by splitting some of its phrases into sub-phrases or by shifting phrase boundaries. This is different from word-based translation models, where a typical assumption is that each target word corresponds to only one source word. As a result of this ambiguity, different segmentations are recruited for different examples during training. That in turn leads to over-fitting which shows in overly determinized estimates of the phrase translation probabilities. In addition, (DeNero et al., 2006) found that the trained phrase table shows a highly peaked distribution in opposition to the more flat distribution resulting from heuristic extraction, leaving the decoder only few translation options at decoding time.

Our work differs from (DeNero et al., 2006) in a number of ways, addressing those problems.

To limit the effects of over-fitting, we apply the leaving-one-out and cross-validation methods in training. In addition, we do not restrict the training to phrases consistent with the word alignment, as was done in (DeNero et al., 2006). This allows us to recover from flawed word alignments.

In (Liang et al., 2006) a discriminative translation system is described. For training of the parameters for the discriminative features they propose a strategy they call *bold updating*. It is similar to our forced alignment training procedure described in Section 3.

For the hierarchical phrase-based approach, (Blunsom et al., 2008) present a discriminative rule model and show the difference between using only the viterbi alignment in training and using the full sum over all possible derivations.

Forced alignment can also be utilized to train a phrase segmentation model, as is shown in (Shen et al., 2008). They report small but consistent improvements by incorporating this segmentation model, which works as an additional prior probability on the monolingual target phrase.

In (Ferrer and Juan, 2009), phrase models are trained by a semi-hidden Markov model. They train a conditional "inverse" phrase model of the target phrase given the source phrase. Additionally to the phrases, they model the segmentation sequence that is used to produce a phrase alignment between the source and the target sentence. They used a phrase length limit of 4 words with longer phrases not resulting in further improvements. To counteract over-fitting, they interpolate the phrase model with IBM Model 1 probabilities that are computed on the phrase level. We also include these word lexica, as they are standard components of the phrase-based system.

It is shown in (Ferrer and Juan, 2009), that Viterbi training produces almost the same results as full Baum-Welch training. They report improvements over a phrase-based model that uses an inverse phrase model and a language model. Experiments are carried out on a custom subset of the English-Spanish Europarl corpus.

Our approach is similar to the one presented in (Ferrer and Juan, 2009) in that we compare Viterbi and a training method based on the Forward-Backward algorithm. But instead of focusing on the statistical model and relaxing the translation task by using monotone translation only, we use a full and competitive translation system as starting point with reordering and all models included.

In (Marcu and Wong, 2002), a joint probability phrase model is presented. The learned phrases are restricted to the most frequent n-grams up to length 6 and all unigrams. Monolingual phrases have to occur at least 5 times to be considered in training. Smoothing is applied to the learned models so that probabilities for rare phrases are non-zero. In training, they use a greedy algorithm to produce the Viterbi phrase alignment and then apply a hill-climbing technique that modifies the Viterbi alignment by merge, move, split, and swap operations to find an alignment with a better probability in each iteration. The model shows improvements in translation quality over the single-word-based IBM Model 4 (Brown et al., 1993) on a subset of the Canadian Hansards corpus.

The joint model by (Marcu and Wong, 2002) is refined by (Birch et al., 2006) who use high-confidence word alignments to constrain the search space in training. They observe that due to several constraints and pruning steps, the trained phrase table is much smaller than the heuristically extracted one, while preserving translation quality.

The work by (DeNero et al., 2008) describes a method to train the joint model described in (Marcu and Wong, 2002) with a Gibbs sampler. They show that by applying a prior distribution over the phrase translation probabilities they can prevent over-fitting. The prior is composed of IBM1 lexical probabilities and a geometric distribution over phrase lengths which penalizes long phrases. The two approaches differ in that we apply the leaving-one-out procedure to avoid over-fitting, as opposed to explicitly defining a prior distribution.

3 Alignment

The training process is divided into three parts. First we obtain all models needed for a normal translations system. We perform minimum error rate training with the downhill simplex algorithm (Nelder and Mead, 1965) on the development data to obtain a set of scaling factors that achieve a good BLEU score. We then use these models and scaling factors to do a forced alignment, where we compute a phrase alignment for the training data. From this alignment we then estimate new phrase models, while keeping all other models un-

changed. In this section we describe our forced alignment procedure that is the basic training procedure for the models proposed here.

3.1 Forced Alignment

The idea of forced alignment is to perform a phrase segmentation and alignment of each sentence pair of the training data using the full translation system as in decoding. What we call segmentation and alignment here corresponds to the "concepts" used by (Marcu and Wong, 2002). We apply our normal phrase-based decoder on the source side of the training data and constrain the translations to the corresponding target sentences from the training data.

Given a source sentence f_1^J and target sentence e_1^I, we search for the best phrase segmentation and alignment that covers both sentences. A segmentation of a sentence into K phrase is defined by

$$k \rightarrow s_k := (i_k, b_k, j_k), \text{ for } k = 1, \ldots, K$$

where for each segment i_k is last position of kth target phrase, and (b_k, j_k) are the start and end positions of the source phrase aligned to the kth target phrase. Consequently, we can modify Equation 2 to define the best segmentation of a sentence pair as:

$$\hat{s}_1^{\hat{K}} = \underset{K, s_1^K}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, s_1^K, f_1^J) \right\} \quad (3)$$

The identical models as in search are used: conditional phrase probabilities $p(\tilde{f}_k|\tilde{e}_k)$ and $p(\tilde{e}_k|\tilde{f}_k)$, within-phrase lexical probabilities, distance-based reordering model as well as word and phrase penalty. A language model is not used in this case, as the system is constrained to the given target sentence and thus the language model score has no effect on the alignment.

In addition to the phrase matching on the source sentence, we also discard all phrase translation candidates, that do not match any sequence in the given target sentence.

Sentences for which the decoder can not find an alignment are discarded for the phrase model training. In our experiments, this is the case for roughly 5% of the training sentences.

3.2 Leaving-one-out

As was mentioned in Section 2, previous approaches found over-fitting to be a problem in phrase model training. In this section, we describe a leaving-one-out method that can improve the phrase alignment in situations, where the probability of rare phrases and alignments might be overestimated. The training data that consists of N parallel sentence pairs f_n and e_n for $n = 1, \ldots, N$ is used for both the initialization of the translation model $p(\tilde{f}|\tilde{e})$ and the phrase model training. While this way we can make full use of the available data and avoid unknown words during training, it has the drawback that it can lead to over-fitting. All phrases extracted from a specific sentence pair f_n, e_n can be used for the alignment of this sentence pair. This includes longer phrases, which only match in very few sentences in the data. Therefore those long phrases are trained to fit only a few sentence pairs, strongly overestimating their translation probabilities and failing to generalize. In the extreme case, whole sentences will be learned as phrasal translations. The average length of the used phrases is an indicator of this kind of over-fitting, as the number of matching training sentences decreases with increasing phrase length. We can see an example in Figure 2. Without leaving-one-out the sentence is segmented into a few long phrases, which are unlikely to occur in data to be translated. Phrase boundaries seem to be unintuitive and based on some hidden structures. With leaving-one-out the phrases are shorter and therefore better suited for generalization to unseen data.

Previous attempts have dealt with the over-fitting problem by limiting the maximum phrase length (DeNero et al., 2006; Marcu and Wong, 2002) and by smoothing the phrase probabilities by lexical models on the phrase level (Ferrer and Juan, 2009). However, (DeNero et al., 2006) experienced similar over-fitting with short phrases due to the fact that the same word sequence can be segmented in different ways, leading to specific segmentations being learned for specific training sentence pairs. Our results confirm these findings. To deal with this problem, instead of simple phrase length restriction, we propose to apply the leaving-one-out method, which is also used for language modeling techniques (Kneser and Ney, 1995).

When using leaving-one-out, we modify the phrase translation probabilities for each sentence pair. For a training example f_n, e_n, we have to remove all phrases $C_n(\tilde{f}, \tilde{e})$ that were extracted from this sentence pair from the phrase counts that

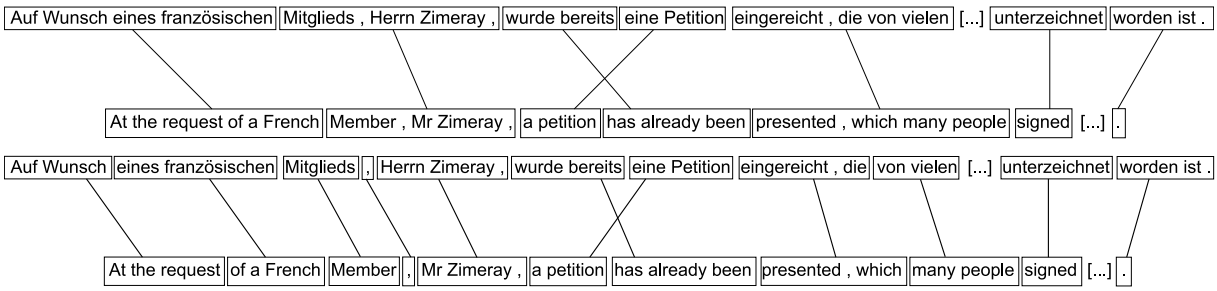


Figure 2: Segmentation example from forced alignment. Top: without leaving-one-out. Bottom: with leaving-one-out.

we used to construct our phrase translation table. The same holds for the marginal counts $C_n(\tilde{e})$ and $C_n(\tilde{f})$. Starting from Equation 1, the leaving-one-out phrase probability for training sentence pair n is

$$p_{l1o,n}(\tilde{f}|\tilde{e}) = \frac{C(\tilde{f},\tilde{e}) - C_n(\tilde{f},\tilde{e})}{C(\tilde{e}) - C_n(\tilde{e})} \quad (4)$$

To be able to perform the re-computation in an efficient way, we store the source and target phrase marginal counts for each phrase in the phrase table. A phrase extraction is performed for each training sentence pair separately using the same word alignment as for the initialization. It is then straightforward to compute the phrase counts after leaving-one-out using the phrase probabilities and marginal counts stored in the phrase table.

While this works well for more frequent observations, singleton phrases are assigned a probability of zero. We refer to singleton phrases as phrase pairs that occur only in one sentence. For these sentences, the decoder needs the singleton phrase pairs to produce an alignment. Therefore we retain those phrases by assigning them a positive probability close to zero. We evaluated with two different strategies for this, which we call standard and length-based leaving-one-out. Standard leaving-one-out assigns a fixed probability α to singleton phrase pairs. This way the decoder will prefer using more frequent phrases for the alignment, but is able to resort to singletons if necessary. However, we found that with this method longer singleton phrases are preferred over shorter ones, because fewer of them are needed to produce the target sentence. In order to better generalize to unseen data, we would like to give the preference to shorter phrases. This is done by length-based leaving-one-out, where singleton phrases are assigned the probability $\beta^{(|\tilde{f}|+|\tilde{e}|)}$ with the source and target phrase lengths $|\tilde{f}|$ and $|\tilde{e}|$ and fixed $\beta < 1$. In our experiments we set $\alpha = e^{-20}$ and $\beta = e^{-5}$. Table 1 shows the decrease in average source phrase length by application of leaving-one-out.

Table 1: Avg. source phrase lengths in forced alignment without leaving-one-out and with standard and length-based leaving-one-out.

	avg. phrase length
without l1o	2.5
standard l1o	1.9
length-based l1o	1.6

3.3 Cross-validation

For the first iteration of the phrase training, leaving-one-out can be implemented efficiently as described in Section 3.2. For higher iterations, phrase counts obtained in the previous iterations would have to be stored on disk separately for each sentence and accessed during the forced alignment process. To simplify this procedure, we propose a cross-validation strategy on larger batches of data. Instead of recomputing the phrase counts for each sentence individually, this is done for a whole batch of sentences at a time. In our experiments, we set this batch-size to 10000 sentences.

3.4 Parallelization

To cope with the runtime and memory requirements of phrase model training that was pointed out by previous work (Marcu and Wong, 2002; Birch et al., 2006), we parallelized the forced alignment by splitting the training corpus into blocks of 10k sentence pairs. From the initial phrase table, each of these blocks only loads the phrases that are required for alignment. The align-

ment and the counting of phrases are done separately for each block and then accumulated to build the updated phrase model.

4 Phrase Model Training

The produced phrase alignment can be given as a single best alignment, as the n-best alignments or as an alignment graph representing all alignments considered by the decoder. We have developed two different models for phrase translation probabilities which make use of the force-aligned training data. Additionally we consider smoothing by different kinds of interpolation of the generative model with the state-of-the-art heuristics.

4.1 Viterbi

The simplest of our generative phrase models estimates phrase translation probabilities by their relative frequencies in the Viterbi alignment of the data, similar to the heuristic model but with counts from the phrase-aligned data produced in training rather than computed on the basis of a word alignment. The translation probability of a phrase pair $(\tilde{f}, \tilde{e})$ is estimated as

$$p_{FA}(\tilde{f}|\tilde{e}) = \frac{C_{FA}(\tilde{f}, \tilde{e})}{\sum_{\tilde{f}'} C_{FA}(\tilde{f}', \tilde{e})} \quad (5)$$

where $C_{FA}(\tilde{f}, \tilde{e})$ is the count of the phrase pair $(\tilde{f}, \tilde{e})$ in the phrase-aligned training data. This can be applied to either the Viterbi phrase alignment or an n-best list. For the simplest model, each hypothesis in the n-best list is weighted equally. We will refer to this model as the *count* model as we simply count the number of occurrences of a phrase pair. We also experimented with weighting the counts with the estimated likelihood of the corresponding entry in the the n-best list. The sum of the likelihoods of all entries in an n-best list is normalized to 1. We will refer to this model as the *weighted count* model.

4.2 Forward-backward

Ideally, the training procedure would consider all possible alignment and segmentation hypotheses. When alternatives are weighted by their posterior probability. As discussed earlier, the run-time requirements for computing all possible alignments is prohibitive for large data tasks. However, we can approximate the space of all possible hypotheses by the search space that was used for the alignment. While this might not cover all phrase translation probabilities, it allows the search space and translation times to be feasible and still contains the most probable alignments. This search space can be represented as a graph of partial hypotheses (Ueffing et al., 2002) on which we can compute expectations using the Forward-Backward algorithm. We will refer to this alignment as the *full* alignment. In contrast to the method described in Section 4.1, phrases are weighted by their posterior probability in the word graph. As suggested in work on minimum Bayes-risk decoding for SMT (Tromble et al., 2008; Ehling et al., 2007), we use a global factor to scale the posterior probabilities.

4.3 Phrase Table Interpolation

As (DeNero et al., 2006) have reported improvements in translation quality by interpolation of phrase tables produced by the generative and the heuristic model, we adopt this method and also report results using log-linear interpolation of the estimated model with the original model.

The log-linear interpolations $p_{int}(\tilde{f}|\tilde{e})$ of the phrase translation probabilities are estimated as

$$p_{int}(\tilde{f}|\tilde{e}) = \Big(p_H(\tilde{f}|\tilde{e})\Big)^{1-\omega} \cdot \Big(p_{gen}(\tilde{f}|\tilde{e})\Big)^{(\omega)} \quad (6)$$

where ω is the interpolation weight, p_H the heuristically estimated phrase model and p_{gen} the count model. The interpolation weight ω is adjusted on the development corpus. When interpolating phrase tables containing different sets of phrase pairs, we retain the intersection of the two.

As a generalization of the fixed interpolation of the two phrase tables we also experimented with adding the two trained phrase probabilities as additional features to the log-linear framework. This way we allow different interpolation weights for the two translation directions and can optimize them automatically along with the other feature weights. We will refer to this method as *feature-wise combination*. Again, we retain the intersection of the two phrase tables. With good log-linear feature weights, feature-wise combination should perform at least as well as fixed interpolation. However, the results presented in Table 5

Table 2: Statistics for the Europarl German-English data

		German	English
TRAIN	Sentences	1 311 815	
	Run. Words	34 398 651	36 090 085
	Vocabulary	336 347	118 112
	Singletons	168 686	47 507
DEV	Sentences	2 000	
	Run. Words	55 118	58 761
	Vocabulary	9 211	6 549
	OOVs	284	77
TEST	Sentences	2 000	
	Run. Words	56 635	60 188
	Vocabulary	9 254	6 497
	OOVs	266	89

show a slightly lower performance. This illustrates that a higher number of features results in a less reliable optimization of the log-linear parameters.

5 Experimental Evaluation

5.1 Experimental Setup

We conducted our experiments on the German-English data published for the *ACL 2008 Workshop on Statistical Machine Translation* (WMT08). Statistics for the Europarl data are given in Table 2.

We are given the three data sets $TRAIN$, DEV and $TEST$. For the heuristic phrase model, we first use GIZA++ (Och and Ney, 2003) to compute the word alignment on $TRAIN$. Next we obtain a phrase table by extraction of phrases from the word alignment. The scaling factors of the translation models have been optimized for BLEU on the DEV data.

The phrase table obtained by heuristic extraction is also used to initialize the training. The forced alignment is run on the training data $TRAIN$ from which we obtain the phrase alignments. Those are used to build a phrase table according to the proposed generative phrase models. Afterward, the scaling factors are trained on DEV for the new phrase table. By feeding back the new phrase table into forced alignment we can reiterate the training procedure. When training is finished the resulting phrase model is evaluated on DEV and $TEST$. Additionally, we can apply smoothing by interpolation of the new phrase table with the original one estimated heuristically, retrain the scaling factors and evaluate afterwards.

Table 3: Comparison of different training setups for the count model on DEV.

leaving-one-out	max phr.len.	BLEU	TER
baseline	6	25.7	61.1
none	2	25.2	61.3
	3	25.7	61.3
	4	25.5	61.4
	5	25.5	61.4
	6	25.4	61.7
standard	6	26.4	60.9
length-based	6	26.5	60.6

The baseline system is a standard phrase-based SMT system with eight features: phrase translation and word lexicon probabilities in both translation directions, phrase penalty, word penalty, language model score and a simple distance-based reordering model. The features are combined in a log-linear way. To investigate the generative models, we replace the two phrase translation probabilities and keep the other features identical to the baseline. For the feature-wise combination the two generative phrase probabilities are added to the features, resulting in a total of 10 features. We used a 4-gram language model with modified Kneser-Ney discounting for all experiments. The metrics used for evaluation are the case-sensitive BLEU (Papineni et al., 2002) score and the translation edit rate (TER) (Snover et al., 2006) with one reference translation.

5.2 Results

In this section, we investigate the different aspects of the models and methods presented before. We will focus on the proposed leaving-one-out technique and show that it helps in finding good phrasal alignments on the training data that lead to improved translation models. Our final results show an improvement of 1.4 BLEU over the heuristically extracted phrase model on the test data set.

In Section 3.2 we have discussed several methods which aim to overcome the over-fitting prob-

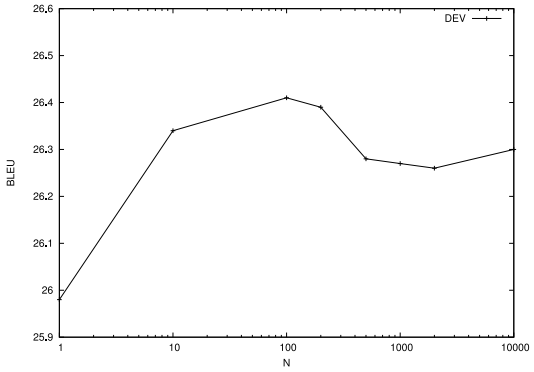


Figure 3: Performance on DEV in BLEU of the count model plotted against size n of n-best list on a logarithmic scale.

lems described in (DeNero et al., 2006). Table 3 shows translation scores of the count model on the development data after the first training iteration for both leaving-one-out strategies we have introduced and for training without leaving-one-out with different restrictions on phrase length. We can see that by restricting the source phrase length to a maximum of 3 words, the trained model is close to the performance of the heuristic phrase model. With the application of leaving-one-out, the trained model is superior to the baseline, the length-based strategy performing slightly better than standard leaving-one-out. For these experiments the count model was estimated with a 100-best list.

The count model we describe in Section 4.1 estimates phrase translation probabilities using counts from the n-best phrase alignments. For smaller n the resulting phrase table contains fewer phrases and is more deterministic. For higher values of n more competing alignments are taken into account, resulting in a bigger phrase table and a smoother distribution. We can see in Figure 3 that translation performance improves by moving from the Viterbi alignment to n-best alignments. The variations in performance with sizes between $n = 10$ and $n = 10000$ are less than 0.2 BLEU. The maximum is reached for $n = 100$, which we used in all subsequent experiments. An additional benefit of the count model is the smaller phrase table size compared to the heuristic phrase extraction. This is consistent with the findings of (Birch et al., 2006). Table 4 shows the phrase table sizes for different n. With $n = 100$ we retain only 17% of the original phrases. Even for the full model, we do not retain all phrase table entries. Due to pruning in the forced alignment step, not all translation options are considered. As a result experiments can be done more rapidly and with less resources than with the heuristically extracted phrase table. Also, our experiments show that the increased performance of the count model is partly derived from the smaller phrase table size. In Table 5 we can see that the performance of the heuristic phrase model can be increased by 0.6 BLEU on $TEST$ by filtering the phrase table to contain the same phrases as the count model and reoptimizing the log-linear model weights. The experiments on the number of different alignments taken into account were done with standard leaving-one-out.

Table 4: Phrase table size of the count model for different n-best list sizes, the full model and for heuristic phrase extraction.

N	# phrases	% of full table
1	4.9M	5.3
10	8.4M	9.1
100	15.9M	17.2
1000	27.1M	29.2
10000	40.1M	43.2
full	59.6M	64.2
heuristic	92.7M	100.0

The final results are given in Table 5. We can see that the count model outperforms the baseline by 0.8 BLEU on DEV and 0.9 BLEU on $TEST$ after the first training iteration. The performance of the filtered baseline phrase table shows that part of that improvement derives from the smaller phrase table size. Application of cross-validation (cv) in the first iteration yields a performance close to training with leaving-one-out (l1o), which indicates that cross-validation can be safely applied to higher training iterations as an alternative to leaving-one-out. The weighted count model clearly under-performs the simpler count model. A second iteration of the training algorithm shows nearly no changes in BLEU score, but a small improvement in TER. Here, we used the phrase table trained with leaving-one-out in the first iteration and applied cross-validation in the second iteration. Log-linear interpolation of the count model with the heuristic yields a further increase, showing an improvement of 1.3 BLEU on DEV and 1.4 BLEU on $TEST$ over the baseline. The interpo-

Table 5: Final results for the heuristic phrase table filtered to contain the same phrases as the count model (baseline filt.), the count model trained with leaving-one-out (l1o) and cross-validation (cv), the weighted count model and the full model. Further, scores for fixed log-linear interpolation of the count model trained with leaving-one-out with the heuristic as well as a feature-wise combination are shown. The results of the second training iteration are given in the bottom row.

	DEV		TEST	
	BLEU	TER	BLEU	TER
baseline	25.7	61.1	26.3	60.9
baseline filt.	26.0	61.6	26.9	61.2
count (l1o)	26.5	60.6	27.2	60.5
count (cv)	26.4	60.7	27.0	60.7
weight. count	25.9	61.4	26.4	61.3
full	26.3	60.0	27.0	60.2
fixed interpol.	**27.0**	**59.4**	**27.7**	**59.2**
feat. comb.	26.8	60.1	27.6	59.9
count, iter. 2	26.4	60.3	27.2	60.0

lation weight is adjusted on the development set and was set to $\omega = 0.6$. Integrating both models into the log-linear framework (feat. comb.) yields a BLEU score slightly lower than with fixed interpolation on both DEV and $TEST$. This might be attributed to deficiencies in the tuning procedure. The full model, where we extract all phrases from the search graph, weighted with their posterior probability, performs comparable to the count model with a slightly worse BLEU and a slightly better TER.

6 Conclusion

We have shown that training phrase models can improve translation performance on a state-of-the-art phrase-based translation model. This is achieved by training phrase translation probabilities in a way that they are consistent with their use in translation. A crucial aspect here is the use of leaving-one-out to avoid over-fitting. We have shown that the technique is superior to limiting phrase lengths and smoothing with lexical probabilities alone.

While models trained from Viterbi alignments already lead to good results, we have demonstrated that considering the 100-best alignments allows to better model the ambiguities in phrase segmentation.

The proposed techniques are shown to be superior to previous approaches that only used lexical probabilities to smooth phrase tables or imposed limits on the phrase lengths. On the WMT08 Europarl task we show improvements of 0.9 BLEU points with the trained phrase table and 1.4 BLEU points when interpolating the newly trained model with the original, heuristically extracted phrase table. In TER, improvements are 0.4 and 1.7 points.

In addition to the improved performance, the trained models are smaller leading to faster and smaller translation systems.

Acknowledgments

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation, and also partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001-06-C-0023. Any opinions, ndings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reect the views of the DARPA.

References

Alexandra Birch, Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Constraining the phrase-based, joint probability statistical translation model. In *smt2006*, pages 154–157, Jun.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio, June. Association for Computational Linguistics.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 25–28, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why Generative Phrase Models Underperform Surface Heuristics. In *Proceedings of the*

Workshop on Statistical Machine Translation, pages 31–38, New York City, June.

John DeNero, Alexandre Buchard-Côté, and Dan Klein. 2008. Sampling Alignment Structure under a Bayesian Translation Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Honolulu, October.

Nicola Ehling, Richard Zens, and Hermann Ney. 2007. Minimum bayes risk decoding for bleu. In *ACL '07: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 101–104, Morristown, NJ, USA. Association for Computational Linguistics.

Jesús-Andrés Ferrer and Alfons Juan. 2009. A phrase-based hidden semi-markov approach to machine translation. In *Procedings of European Association for Machine Translation (EAMT)*, Barcelona, Spain, May. European Association for Machine Translation.

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-Off for M-gram Language Modelling. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–184, Detroit, MI, May.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Percy Liang, Alexandre Buchard-Côté, Dan Klein, and Ben Taskar. 2006. An End-to-End Discriminative Approach to Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 761–768, Sydney, Australia.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, July.

J.A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *The Computer Journal)*, 7:308–313.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, December.

F.J. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP99)*, pages 20–28, University of Maryland, College Park, MD, USA, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Wade Shen, Brian Delaney, Tim Anderson, and Ray Slyh. 2008. The MIT-LL/AFRL IWSLT-2008 MT System. In *Proceedings of IWSLT 2008*, pages 69–76, Hawaii, U.S.A., October.

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

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629, Honolulu, Hawaii, October. Association for Computational Linguistics.

N. Ueffing, F.J. Och, and H. Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing*, pages 156–163, Philadelphia, PA, USA, July.

Efficient Staggered Decoding for Sequence Labeling

Nobuhiro Kaji Yasuhiro Fujiwara Naoki Yoshinaga Masaru Kitsuregawa
Institute of Industrial Science,
The University of Tokyo,
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505 Japan
{kaji,fujiwara,ynaga,kisture}@tkl.iis.u-tokyo.ac.jp

Abstract

The Viterbi algorithm is the conventional decoding algorithm most widely adopted for sequence labeling. Viterbi decoding is, however, prohibitively slow when the label set is large, because its time complexity is quadratic in the number of labels. This paper proposes an exact decoding algorithm that overcomes this problem. A novel property of our algorithm is that it efficiently reduces the labels to be decoded, while still allowing us to check the optimality of the solution. Experiments on three tasks (POS tagging, joint POS tagging and chunking, and supertagging) show that the new algorithm is several orders of magnitude faster than the basic Viterbi and a state-of-the-art algorithm, CARPEDIEM (Esposito and Radicioni, 2009).

1 Introduction

In the past decade, sequence labeling algorithms such as HMMs, CRFs, and Collins' perceptrons have been extensively studied in the field of NLP (Rabiner, 1989; Lafferty et al., 2001; Collins, 2002). Now they are indispensable in a wide range of NLP tasks including chunking, POS tagging, NER and so on (Sha and Pereira, 2003; Tsuruoka and Tsujii, 2005; Lin and Wu, 2009).

One important task in sequence labeling is how to find the most probable label sequence from among all possible ones. This task, referred to as decoding, is usually carried out using the Viterbi algorithm (Viterbi, 1967). The Viterbi algorithm has $O(NL^2)$ time complexity,[1] where N is the input size and L is the number of labels. Although the Viterbi algorithm is generally efficient, it becomes prohibitively slow when dealing with a large number of labels, since its computational cost is quadratic in L (Dietterich et al., 2008).

Unfortunately, several sequence-labeling problems in NLP involve a large number of labels. For example, there are more than 40 and 2000 labels in POS tagging and supertagging, respectively (Brants, 2000; Matsuzaki et al., 2007). These tasks incur much higher computational costs than simpler tasks like NP chunking. What is worse, the number of labels grows drastically if we jointly perform multiple tasks. As we shall see later, we need over 300 labels to reduce joint POS tagging and chunking into the single sequence labeling problem. Although joint learning has attracted much attention in recent years, how to perform decoding efficiently still remains an open problem.

In this paper, we present a new decoding algorithm that overcomes this problem. The proposed algorithm has three distinguishing properties: (1) It is much more efficient than the Viterbi algorithm when dealing with a large number of labels. (2) It is an exact algorithm, that is, the optimality of the solution is always guaranteed unlike approximate algorithms. (3) It is automatic, requiring no task-dependent hyperparameters that have to be manually adjusted.

Experiments evaluate our algorithm on three tasks: POS tagging, joint POS tagging and chunking, and supertagging[2]. The results demonstrate that our algorithm is up to several orders of magnitude faster than the basic Viterbi algorithm and a state-of-the-art algorithm (Esposito and Radicioni, 2009); it makes exact decoding practical even in labeling problems with a large label set.

2 Preliminaries

We first provide a brief overview of sequence labeling and introduce related work.

[1] The first-order Markov assumption is made throughout this paper, although our algorithm is applicable to higher-order Markov models as well.

[2] Our implementation is available at http://www.tkl.iis.u-tokyo.ac.jp/~kaji/staggered

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 485–494,
Uppsala, Sweden, 11-16 July 2010.

2.1 Models

Sequence labeling is the problem of predicting label sequence $\boldsymbol{y} = \{y_n\}_{n=1}^N$ for given token sequence $\boldsymbol{x} = \{x_n\}_{n=1}^N$. This is typically done by defining a score function $f(\boldsymbol{x}, \boldsymbol{y})$ and locating the best label sequence: $\boldsymbol{y}_{max} = \operatorname{argmax}_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$. The form of $f(\boldsymbol{x}, \boldsymbol{y})$ is dependent on the learning model used. Here, we introduce two models widely used in the literature.

Generative models HMM is the most famous generative model for labeling token sequences (Rabiner, 1989). In HMMs, the score function $f(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution over $(\boldsymbol{x}, \boldsymbol{y})$. If we assume a one-to-one correspondence between the hidden states and the labels, the score function can be written as:

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{y}) &= \log p(\boldsymbol{x}, \boldsymbol{y}) \\ &= \log p(\boldsymbol{x}|\boldsymbol{y}) + \log p(\boldsymbol{y}) \\ &= \sum_{n=1}^{N} \log p(x_n|y_n) + \sum_{n=1}^{N} \log p(y_n|y_{n-1}). \end{aligned}$$

The parameters $\log p(x_n|y_n)$ and $\log p(y_n|y_{n-1})$ are usually estimated using maximum likelihood or the EM algorithm. Since parameter estimation lies outside the scope of this paper, a detailed description is omitted.

Discriminative models Recent years have seen the emergence of discriminative training methods for sequence labeling (Lafferty et al., 2001; Tasker et al., 2003; Collins, 2002; Tsochantaridis et al., 2005). Among them, we focus on the perceptron algorithm (Collins, 2002). Although we do not discuss the other discriminative models, our algorithm is equivalently applicable to them. The major difference between those models lies in parameter estimation; the decoding process is virtually the same.

In the perceptron, the score function $f(\boldsymbol{x}, \boldsymbol{y})$ is given as $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{w}$ is the weight vector, and $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ is the feature vector representation of the pair $(\boldsymbol{x}, \boldsymbol{y})$. By making the first-order Markov assumption, we have

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} w_k \phi_k(\boldsymbol{x}, y_{n-1}, y_n), \end{aligned}$$

where $K = |\phi(\boldsymbol{x}, \boldsymbol{y})|$ is the number of features, ϕ_k is the k-th feature function, and w_k is the weight corresponding to it. Parameter $\boldsymbol{w}$ can be estimated in the same way as in the conventional perceptron algorithm. See (Collins, 2002) for details.

2.2 Viterbi decoding

Given the score function $f(\boldsymbol{x}, \boldsymbol{y})$, we have to locate the best label sequence. This is usually performed by applying the Viterbi algorithm. Let $\omega(y_n)$ be the best score of the partial label sequence ending with y_n. The idea of the Viterbi algorithm is to use dynamic programming to compute $\omega(y_n)$. In HMMs, $\omega(y_n)$ can be can be defined as

$$\max_{y_{n-1}} \{\omega(y_{n-1}) + \log p(y_n|y_{n-1})\} + \log p(x_n|y_n).$$

Using this recursive definition, we can evaluate $\omega(y_n)$ for all y_n. This results in the identification of the best label sequence.

Although the Viterbi algorithm is commonly adopted in past studies, it is not always efficient. The computational cost of the Viterbi algorithm is $O(NL^2)$, where N is the input length and L is the number of labels; it is efficient enough if L is small. However, if there are many labels, the Viterbi algorithm becomes prohibitively slow because of its quadratic dependence on L.

2.3 Related work

To the best of our knowledge, the Viterbi algorithm is the only algorithm widely adopted in the NLP field that offers exact decoding. In other communities, several exact algorithms have already been proposed for handling large label sets. While they are successful to some extent, they demand strong assumptions that are unusual in NLP. Moreover, none were challenged with standard NLP tasks.

Felzenszwalb et al. (2003) presented a fast inference algorithm for HMMs based on the assumption that the hidden states can be embedded in a grid space, and the transition probability corresponds to the distance on that space. This type of probability distribution is not common in NLP tasks. Lifshits et al. (2007) proposed a compression-based approach to speed up HMM decoding. It assumes that the input sequence is highly repetitive. Amongst others, CARPEDIEM (Esposito and Radicioni, 2009) is the algorithm closest to our work. It accelerates decoding by assuming that the adjacent labels are not strongly correlated. This assumption is appropriate for

some NLP tasks. For example, as suggested in (Liang et al., 2008), adjacent labels do not provide strong information in POS tagging. However, the applicability of this idea to other NLP tasks is still unclear.

Approximate algorithms, such as beam search or island-driven search, have been proposed for speeding up decoding. Tsuruoka and Tsujii (2005) proposed easiest-first deterministic decoding. Siddiqi and Moore (2005) presented the parameter tying approach for fast inference in HMMs. A similar idea was applied to CRFs as well (Cohn, 2006; Jeong et al., 2009).

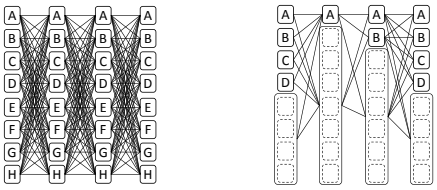
In general, approximate algorithms have the advantage of speed over exact algorithms. However, both types of algorithms are still widely adopted by practitioners, since exact algorithms have merits other than speed. First, the optimality of the solution is always guaranteed. It is hard for most of the approximate algorithms to even bound the error rate. Second, approximate algorithms usually require hyperparameters, which control the trade-off between accuracy and efficiency (e.g., beam width), and these have to be manually adjusted. On the other hand, most of the exact algorithms, including ours, do not require such a manual effort.

Despite these advantages, exact algorithms are rarely used when dealing with a large number of labels. This is because exact algorithms become considerably slower than approximate algorithms in such situations. The paper presents an exact algorithm that avoids this problem; it provides the research community with another option for handling a lot of labels.

3 Algorithm

This section presents the new decoding algorithm. The key is to reduce the number of labels examined. Our algorithm locates the best label sequence by iteratively solving labeling problems with a reduced label set. This results in significant time savings in practice, because each iteration becomes much more efficient than solving the original labeling problem. More importantly, our algorithm always obtains the exact solution. This is because the algorithm allows us to check the optimality of the solution achieved by using only the reduced label set.

In the following discussions, we restrict our focus to HMMs for presentation clarity. Extension to the perceptron algorithm is presented in Section 4.



(a) (b)

Figure 1: (a) An example of a lattice, where the letters {A, B, C, D, E, F, G, H} represent labels associated with nodes. (b) The degenerate lattice.

3.1 Degenerate lattice

We begin by introducing the *degenerate lattice*, which plays a central role in our algorithm. Consider the lattice in Figure 1(a). Following convention, we regard each path on the lattice as a label sequence. Note that the label set is {A, B, C, D, E, F, G, H}. By aggregating several nodes in the same column of the lattice, we can transform the original lattice into a simpler form, which we call the degenerate lattice (Figure 1(b)).

Let us examine the intuition behind the degenerate lattice. Aggregating nodes can be viewed as grouping several labels into a new one. Here, a label is referred to as an *active label* if it is not aggregated (e.g., A, B, C, and D in the first column of Figure 1(b)), and otherwise as an *inactive label* (i.e., dotted nodes). The new label, which is made by grouping the inactive labels, is referred to as a *degenerate label* (i.e., large nodes covering the dotted ones). Two degenerate labels can be seen as equivalent if their corresponding inactive label sets are the same (e.g., degenerate labels in the first and the last column). In this approach, each path of the degenerate lattice can also be interpreted as a label sequence. In this case, however, the label to be assigned is either an active label or a degenerate label.

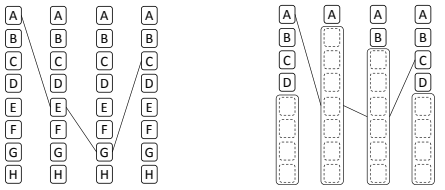
We then define the parameters associated with degenerate label z. For reasons that will become clear later, they are set to the maxima among the parameters of the inactive labels:

$$\log p(x|z) = \max_{y' \in I(z)} \log p(x|y'), \quad (1)$$

$$\log p(z|y) = \max_{y' \in I(z)} \log p(y'|y), \quad (2)$$

$$\log p(y|z) = \max_{y' \in I(z)} \log p(y|y'), \quad (3)$$

$$\log p(z|z') = \max_{y' \in I(z), y'' \in I(z')} \log p(y'|y''), \quad (4)$$



(a) (b)

Figure 2: (a) The path $\boldsymbol{y}$ = {A, E, G, C} of the original lattice. (b) The path $\boldsymbol{z}$ of the degenerate lattice that corresponds to $\boldsymbol{y}$.

where y is an active label, z and z' are degenerate labels, and $I(z)$ denotes one-to-one mapping from z to its corresponding inactive label set.

The degenerate lattice has an important property which is the key to our algorithm:

Lemma 1. *If the best path of the degenerate lattice does not include any degenerate label, it is equivalent to the best path of the original lattice.*

Proof. Let $\boldsymbol{z}_{max}$ be the best path of the degenerate lattice. Our goal is to prove that if $\boldsymbol{z}_{max}$ does not include any degenerate label, then

$$\forall \boldsymbol{y} \in Y, \quad \log p(\boldsymbol{x}, \boldsymbol{y}) \leq \log p(\boldsymbol{x}, \boldsymbol{z}_{max}) \qquad (5)$$

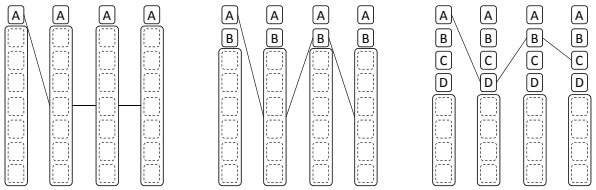
where Y is the set of all paths on the original lattice. We prove this by partitioning Y into two disjoint sets: Y_0 and Y_1, where Y_0 is the subset of Y appearing in the degenerate lattice. Notice that $\boldsymbol{z}_{max} \in Y_0$. Since $\boldsymbol{z}_{max}$ is the best path of the degenerate lattice, we have

$$\forall \boldsymbol{y} \in Y_0, \quad \log p(\boldsymbol{x}, \boldsymbol{y}) \leq \log p(\boldsymbol{x}, \boldsymbol{z}_{max}).$$

The equation holds when $\boldsymbol{y} = \boldsymbol{z}_{max}$. We next examine the label sequence $\boldsymbol{y}$ such that $\boldsymbol{y} \in Y_1$. For each path $\boldsymbol{y} \in Y_1$, there exists a unique path $\boldsymbol{z}$ on the degenerate lattice that corresponds to $\boldsymbol{y}$ (Figure 2). Therefore, we have

$$\forall \boldsymbol{y} \in Y_1, \ \exists \boldsymbol{z} \in Z, \ \log p(\boldsymbol{x}, \boldsymbol{y}) \leq \log p(\boldsymbol{x}, \boldsymbol{z}) < \log p(\boldsymbol{x}, \boldsymbol{z}_{max})$$

where Z is the set of all paths of the degenerate lattice. The inequality $\log p(\boldsymbol{x}, \boldsymbol{y}) \leq \log p(\boldsymbol{x}, \boldsymbol{z})$ can be proved by using Equations (1)-(4). Using these results, we can complete (5). □



(a) (b) (c)

Figure 3: (a) The best path of the initial degenerate lattice, which is denoted by the line, is located. (b) The active labels are expanded and the best path is searched again. (c) The best path without degenerate labels is obtained.

3.2 Staggered decoding

Now we can describe our algorithm, which we call *staggered decoding*. The algorithm successively constructs degenerate lattices and checks whether the best path includes degenerate labels. In building each degenerate lattice, labels with high probability $p(y)$, estimated from training data, are preferentially selected as the active label; the expectation is that such labels are likely to belong to the best path. The algorithm is detailed as follows:

Initialization step The algorithm starts by building a degenerate lattice in which there is only one active label in each column. We select label y with the highest $p(y)$ as the active label.

Search step The best path of the degenerate lattice is located (Figure 3(a)). This is done by using the Viterbi algorithm (and pruning technique, as we describe in Section 3.3). If the best path does not include any degenerate label, we can terminate the algorithm since it is identical with the best path of the original lattice according to Lemma 1. Otherwise, we proceed to the next step.

Expansion step We double the number of the active labels in the degenerate lattice. The new active labels are selected from the current inactive label set in descending order of $p(y)$. If the inactive label set becomes empty, we simply reconstructed the original lattice. After expanding the active labels, we go back to the previous step (Figure 3(b)). This procedure is repeated until the termination condition in the search step is satisfied, i.e., the best path has no degenerate label (Figure 3(c)).

Compared to the Viterbi algorithm, staggered decoding requires two additional computations for

training. First, we have to estimate $p(y)$ so as to select active labels in the initialization and expansion step. Second, we have to compute the parameters regarding degenerate labels according to Equations (1)-(4). Both impose trivial computation costs.

3.3 Pruning

To achieve speed-up, it is crucial that staggered decoding efficiently performs the search step. For this purpose, we can basically use the Viterbi algorithm. In earlier iterations, the Viterbi algorithm is indeed efficient because the label set to be handled is much smaller than the original one. In later iterations, however, our algorithm drastically increases the number of labels, making Viterbi decoding quite expensive.

To handle this problem, we propose a method of pruning the lattice nodes. This technique is motivated by the observation that the degenerate lattice shares many active labels with the previous iteration. In the remainder of Section3.3, we explain the technique by taking the following steps:

- Section 3.3.1 examines a lower bound l such that $l \leq \max_{\boldsymbol{y}} \log p(\boldsymbol{x}, \boldsymbol{y})$.
- Section 3.3.2 examines the maximum score $\textsc{Max}(y_n)$ in case token x_n takes label y_n:

$$\textsc{Max}(y_n) = \max_{y'_n = y_n} \log p(\boldsymbol{x}, \boldsymbol{y}').$$

- Section 3.3.3 presents our pruning procedure. The idea is that if $\textsc{Max}(y_n) < l$, then the node corresponding to y_n can be removed from consideration.

3.3.1 Lower bound

Lower bound l can be trivially calculated in the search step. This can be done by retaining the best path among those consisting of only active labels. The score of that path is obviously the lower bound. Since the search step is repeated until the termination criteria is met, we can update the lower bound at every search step. As the iteration proceeds, the degenerate lattice becomes closer to the original one, so the lower bound becomes tighter.

3.3.2 Maximum score

The maximum score $\textsc{Max}(y_n)$ can be computed from the original lattice. Let $\omega(y_n)$ be the best score of the partial label sequence ending with y_n. Presuming that we traverse the lattice from left to right, $\omega(y_n)$ can be defined as

$$\max_{y_{n-1}} \{\omega(y_{n-1}) + \log p(y_n | y_{n-1})\} + \log p(x_n | y_n).$$

If we traverse the lattice from right to left, an analogous score $\bar{\omega}(y_n)$ can be defined as

$$\log p(x_n | y_n) + \max_{y_{n+1}} \{\bar{\omega}(y_{n+1}) + \log p(y_n | y_{n+1})\}.$$

Using these two scores, we have

$$\textsc{Max}(y_n) = \omega(y_n) + \bar{\omega}(y_n) - \log p(x_n | y_n).$$

Notice that updating $\omega(y_n)$ or $\bar{\omega}(y_n)$ is equivalent to the forward or backward Viterbi algorithm, respectively.

Although it is expensive to compute $\omega(y_n)$ and $\bar{\omega}(y_n)$, we can efficiently estimate their upper bounds. Let $\lambda(y_n)$ and $\bar{\lambda}(y_n)$ be scores analogous to $\omega(y_n)$ and $\bar{\omega}(y_n)$ that are computed using the degenerate lattice. We have $\omega(y_n) \leq \lambda(y_n)$ and $\bar{\omega}(y_n) \leq \bar{\lambda}(y_n)$, by following similar discussions as raised in the proof of Lemma 1. Therefore, we can still check whether $\textsc{Max}(y_n)$ is smaller than l by using $\lambda(y_n)$ and $\bar{\lambda}(y_n)$:

$$\begin{aligned} \textsc{Max}(y_n) &= \omega(y_n) + \bar{\omega}(y_n) - \log p(x_n | y_n) \\ &\leq \lambda(y_n) + \bar{\lambda}(y_n) - \log p(x_n | y_n) \\ &< l. \end{aligned}$$

For the sake of simplicity, we assume that y_n is an active label. Although we do not discuss the other cases, our pruning technique is also applicable to them. We just point out that, if y_n is an inactive label, then there exists a degenerate label z_n in the n-th column such that $y_n \in I(z_n)$, and we can use $\lambda(z_n)$ and $\bar{\lambda}(z_n)$ instead of $\lambda(y_n)$ and $\bar{\lambda}(y_n)$.

We compute $\lambda(y_n)$ and $\bar{\lambda}(y_n)$ by using the forward and backward Viterbi algorithm, respectively. In the search step immediately following initialization, we perform the forward Viterbi algorithm to find the best path, that is, $\lambda(y_n)$ is updated for all y_n. In the next search step, the backward Viterbi algorithm is carried out, and $\bar{\lambda}(y_n)$ is updated. In the succeeding search steps, these updates are alternated. As the algorithm progresses, $\lambda(y_n)$ and $\bar{\lambda}(y_n)$ become closer to $\omega(y_n)$ and $\bar{\omega}(y_n)$.

3.3.3 Pruning procedure

We make use of the bounds in pruning the lattice nodes. To do this, we keep the values of l, $\lambda(y_n)$

and $\bar{\lambda}(y_n)$. They are set as $l = -\infty$ and $\lambda(y_n) = \bar{\lambda}(y_n) = \infty$ in the initialization step, and are updated in the search step. The lower bound l is updated at the end of the search step, while $\lambda(y_n)$ and $\bar{\lambda}(y_n)$ can be updated during the running of the Viterbi algorithm. When $\lambda(y_n)$ or $\bar{\lambda}(y_n)$ is changed, we check whether $\textsc{Max}(y_n) < l$ holds and the node is pruned if the condition is met.

3.4 Analysis

We provide here a theoretical analysis of staggered decoding. In the following proofs, L, V, and N represent the number of original labels, the number of distinct tokens, and the length of input token sequence, respectively. To simplify the discussion, we assume that $\log_2 L$ is an integer (e.g., $L = 64$).

We first introduce three lemmas:

Lemma 2. *Staggered decoding requires at most* $(\log_2 L + 1)$ *iterations to terminate.*

Proof. We have 2^{m-1} active labels in the m-th search step ($m = 1, 2 \ldots$), which means we have L active labels and no degenerate labels in the $(\log_2 L + 1)$-th search step. Therefore, the algorithm always terminates within $(\log_2 L + 1)$ iterations. □

Lemma 3. *The number of degenerate labels is* $\log_2 L$.

Proof. Since we create one new degenerate label in all but the last expansion step, we have $\log_2 L$ degenerate labels. □

Lemma 4. *The Viterbi algorithm requires* $O(L^2 + LV)$ *memory space and has* $O(NL^2)$ *time complexity.*

Proof. Since we need $O(L^2)$ and $O(LV)$ space to keep the transition and emission probability matrices, we need $O(L^2 + LV)$ space to perform the Viterbi algorithm. The time complexity of the Viterbi algorithm is $O(NL^2)$ since there are NL nodes in the lattice and it takes $O(L)$ time to evaluate the score of each node. □

The above statements allow us to establish our main results:

Theorem 1. *Staggered decoding requires* $O(L^2 + LV)$ *memory space.*

Proof. Since we have L original labels and $\log_2 L$ degenerate labels, staggered decoding requires $O((L+\log_2 L)^2 + (L+\log_2 L)V) = O(L^2 + LV)$ memory space to perform Viterbi decoding in the search step. □

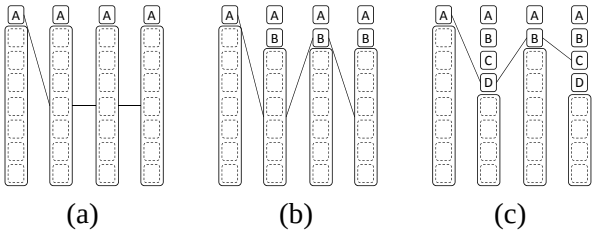


Figure 4: Staggered decoding with column-wise expansion: (a) The best path of the initial degenerate lattice, which does not pass through the degenerate label in the first column. (b) Column-wise expansion is performed and the best path is searched again. Notice that the active label in the first column is not expanded. (c) The final result.

Theorem 2. *Staggered decoding has* $O(N)$ *best case time complexity and* $O(NL^2)$ *worst case time complexity.*

Proof. To perform the m-th search step, staggered decoding requires the order of $O(N4^{m-1})$ time because we have 2^{m-1} active labels. Therefore, it has $O(\sum_{m=1}^{M} N4^{m-1})$ time complexity if it terminates after the M-th search step. In the best case, $M = 1$, the time complexity is $O(N)$. In the worst case, $M = \log_2 L + 1$, the time complexity is the order of $O(NL^2)$ because $\sum_{m=1}^{\log_2 L+1} N4^{m-1} < \frac{4}{3}NL^2$. □

Theorem 1 shows that staggered decoding asymptotically requires the same order of memory space as the Viterbi algorithm. Theorem 2 reveals that staggered decoding has the same order of time complexity as the Viterbi algorithm even in the worst case.

3.5 Heuristic techniques

We present two heuristic techniques for further speeding up our algorithm.

First, we can initialize the value of lower bound l by selecting a path from the original lattice in some way, and then computing the score of that path. In our experiments, we use the path located by the left-to-right deterministic decoding (i.e., beam search with a beam width of 1). Although this method requires an additional cost to locate the path, it is very effective in practice. If l is initialized in this manner, the best case time complexity of our algorithm becomes $O(NL)$.

The second technique is for the expansion step. Instead of the expansion technique described in Section 3.2, we can expand the active labels in a heuristic manner to keep the number of active labels small:

Column-wise expansion step We double the number of the active labels in the column only if the best path of the degenerate lattice passes through the degenerate label of that column (Figure 4).

A drawback of this strategy is that the algorithm requires $N(\log_2 L+1)$ iterations in the worst case. As the result, we can no longer derive a reasonable upper bound for the time complexity. Nevertheless, column-wise expansion is highly effective in practice as we will demonstrate in the experiment. Note that Theorem 1 still holds true even if we use column-wise expansion.

4 Extension to the Perceptron

The discussion we have made so far can be applied to perceptrons. This can be clarified by comparing the score functions $f(\boldsymbol{x}, \boldsymbol{y})$. In HMMs, the score function can be written as

$$\sum_{n=1}^{N} \Big\{ \log(x_n|y_n) + \log(y_n|y_{n-1}) \Big\}.$$

In perceptrons, on the other hand, it is given as

$$\sum_{n=1}^{N} \Big\{ \sum_k w_k^1 \phi_k^1(\boldsymbol{x}, y_n) + \sum_k w_k^2 \phi_k^2(\boldsymbol{x}, y_{n-1}, y_n) \Big\}$$

where we explicitly distinguish the unigram feature function ϕ_k^1 and bigram feature function ϕ_k^2. Comparing the form of the two functions, we can see that our discussion on HMMs can be extended to perceptrons by substituting $\sum_k w_k^1 \phi_k^1(\boldsymbol{x}, y_n)$ and $\sum_k w_k^2 \phi_k^2(\boldsymbol{x}, y_{n-1}, y_n)$ for $\log p(x_n|y_n)$ and $\log p(y_n|y_{n-1})$.

However, implementing the perceptron algorithm is not straightforward. The problem is that it is difficult, if not impossible, to compute $\sum_k w_k^1 \phi_k^1(\boldsymbol{x}, y)$ and $\sum_k w_k^2 \phi_k^2(\boldsymbol{x}, y, y')$ offline because they are dependent on the entire token sequence $\boldsymbol{x}$, unlike $\log p(x|y)$ and $\log p(y|y')$. Consequently, we cannot evaluate the maxima analogous to Equations (1)-(4) offline either.

For unigram features, we compute the maximum, $\max_y \sum_k w_k^1 \phi_k^1(\boldsymbol{x}, y)$, as a preprocess in the initialization step (cf. Equation (1)). This preprocess requires $O(NL)$ time, which is negligible compared with the cost required by the Viterbi algorithm.

Unfortunately, we cannot use the same technique for computing $\max_{y,y'} \sum_k w_k^2 \phi_k^2(\boldsymbol{x}, y, y')$ because a similar computation would take $O(NL^2)$ time (cf. Equation (4)). For bigram features, we compute its upper bound offline. For example, the following bound was proposed by Esposito and Radicioni (2009):

$$\max_{y,y'} \sum_k w_k^2 \phi_k^2(\boldsymbol{x}, y, y') \leq \max_{y,y'} \sum_k w_k^2 \delta(0 < w_k^2)$$

where $\delta(\cdot)$ is the delta function and the summations are taken over all feature functions associated with both y and y'. Intuitively, the upper bound corresponds to an ideal case in which all features with positive weight are activated.[3] It can be computed without any task-specific knowledge.

In practice, however, we can compute better bounds based on task-specific knowledge. The simplest case is that the bigram features are independent of the token sequence $\boldsymbol{x}$. In such a situation, we can trivially compute the exact maxima offline, as we did in the case of HMMs. Fortunately, such a feature set is quite common in NLP problems and we could use this technique in our experiments. Even if bigram features are dependent on $\boldsymbol{x}$, it is still possible to compute better bounds if several features are mutually exclusive, as discussed in (Esposito and Radicioni, 2009).

Finally, it is worth noting that we can use staggered decoding in training perceptrons as well, although such application lies outside the scope of this paper. The algorithm does not support training acceleration for other discriminative models.

5 Experiments and Discussion

5.1 Setting

The proposed algorithm was evaluated with three tasks: POS tagging, joint POS tagging and chunking (called joint tagging for short), and supertagging. To reduce joint tagging into a single sequence labeling problem, we produced the labels by concatenating the POS tag and the chunk tag (BIO format), e.g., NN/B-NP. In the two tasks other than supertagging, the input token is the word. In supertagging, the token is the pair of the word and its oracle POS tag.

[3] We assume binary feature functions.

Table 1: Decoding speed (sent./sec).

	POS tagging	Joint tagging	Supertagging
VITERBI	4000	77	1.1
CARPEDIEM	8600	51	0.26
SD	8800	850	121
SD+C-EXP.	**14,000**	**1600**	**300**

Table 2: The average number of iterations.

	POS tagging	Joint tagging	Supertagging
SD	6.02	8.15	10.0
SD+C-EXP.	6.12	8.62	10.6

Table 3: Training time.

	POS tagging	Joint tagging	Supertagging
VITERBI	100 sec.	20 min.	100 hour
SD+C-EXP.	37 sec.	1.5 min.	5.3 hour

The data sets we used for the three experiments are the Penn TreeBank (PTB) corpus, CoNLL 2000 corpus, and an HPSG treebank built from the PTB corpus (Matsuzaki et al., 2007). We used sections 02-21 of PTB for training, and section 23 for testing. The number of labels in the three tasks is 45, 319 and 2602, respectively.

We used the perceptron algorithm for training. The models were averaged over 10 iterations (Collins, 2002). For features, we basically followed previous studies (Tsuruoka and Tsujii, 2005; Sha and Pereira, 2003; Ninomiya et al., 2006). In POS tagging, we used unigrams of the current and its neighboring words, word bigrams, prefixes and suffixes of the current word, capitalization, and tag bigrams. In joint tagging, we also used the same features. In supertagging, we used POS unigrams and bigrams in addition to the same features other than capitalization.

As the evaluation measure, we used the average decoding speed (sentences/sec) to two significant digits over five trials. To strictly measure the time spent for decoding, we ignored the preprocessing time, that is, the time for loading the model file and converting the features (i.e., strings) into integers. We note that the accuracy was comparable to the state-of-the-art in the three tasks: 97.08, 93.21, and 91.20% respectively.

5.2 Results and discussions

Table 1 presents the performance of our algorithm. SD represents the proposed algorithm without column-wise expansion, while SD+C-EXP. uses column-wise expansion. For comparison, we present the results of two baseline algorithms as well: VITERBI and CARPEDIEM (Esposito and Radicioni, 2009). In almost all settings, we see that both of our algorithms outperformed the other two. We also find that SD+C-EXP. performed consistently better than SD. This indicates the effectiveness of column-wise expansion.

Following VITERBI, CARPEDIEM is the most relevant algorithm, for sequence labeling in NLP, as discussed in Section 2.3. However, our results demonstrated that CARPEDIEM worked poorly in two of the three tasks. We consider this is because the transition information is crucial for the two tasks, and the assumption behind CARPEDIEM is violated. In contrast, the proposed algorithms performed reasonably well for all three tasks, demonstrating the wide applicability of our algorithm.

Table 2 presents the average iteration numbers of SD and SD+C-EXP. We can observe that the two algorithms required almost the same number of iterations on average, although the iteration number is not tightly bounded if we use column-wise expansion. This indicates that SD+C-EXP. virtually avoided performing extra iterations, while heuristically restricting active label expansion.

Table 3 compares the training time spent by VITERBI and SD+C-EXP. Although speeding up perceptron training is a by-product, it is interesting to see that our algorithm is in fact effective at reducing the training time as well. The result also indicates that the speed-up is more significant at test time. This is probably because the model is not predictive enough at the beginning of training, and the pruning is not that effective.

5.3 Comparison with approximate algorithm

Table 4 compares two exact algorithms (VITERBI and SD+E-XP.) with beam search, which is the approximate algorithm widely adopted for sequence labeling in NLP. For this experiment, the beam width, B, was exhaustively calibrated: we tried B = $\{1, 2, 4, 8, ...\}$ until the beam search achieved comparable accuracy to the exact algorithms, i.e., the difference fell below 0.1 in our case.

We see that there is a substantial difference in the performance between VITERBI and BEAM. On the other hand, SD+C-EXP. reached speeds very close to those of BEAM. In fact, they achieved comparable performance in our experiment. These results demonstrate that we could successfully bridge the gap in the performance be-

Table 4: Comparison with beam search (sent./sec).

	POS tagging	Joint tagging	Supertagging
VITERBI	4000	77	1.1
SD+C-EXP.	14,000	1600	**300**
BEAM	**18,000**	**2400**	180

tween exact and approximate algorithms, while retaining the advantages of exact algorithms.

6 Relation to coarse-to-fine approach

Before concluding remarks, we briefly examine the relationship between staggered decoding and coarse-to-fine PCFG parsing (2006). In coarse-to-fine parsing, the candidate parse trees are pruned by using the parse forest produced by a coarse-grained PCFG. Since the degenerate label can be interpreted as a coarse-level label, one may consider that staggered decoding is an instance of coarse-to-fine approach. While there is some resemblance, there are at least two essential differences. First, coarse-to-fine approach is a heuristic pruning, that is, it is not an exact algorithm. Second, our algorithm does not always perform decoding at the fine-grained level. It is designed to be able to stop decoding at the coarse-level.

7 Conclusions

The sequence labeling algorithm is indispensable to modern statistical NLP. However, the Viterbi algorithm, which is the standard decoding algorithm in NLP, is not efficient when we have to deal with a large number of labels. In this paper we presented staggered decoding, which provides a principled way of resolving this problem. We consider that it is a real alternative to the Viterbi algorithm in various NLP tasks.

An interesting future direction is to extend the proposed technique to handle more complex structures than the Markov chains, including semi-Markov models and factorial HMMs (Sarawagi and Cohen, 2004; Sutton et al., 2004). We hope this work opens a new perspective on decoding algorithms for a wide range of NLP problems, not just sequence labeling.

Acknowledgement

We wish to thank the anonymous reviewers for their helpful comments, especially on the computational complexity of our algorithm. We also thank Yusuke Miyao for providing us with the HPSG Treebank data.

References

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of ANLP*, pages 224–231.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multi-level coarse-to-fine PCFG parsing. In *Proceedings of NAACL*, pages 168–175.

Trevor Cohn. 2006. Efficient inference in large conditional random fields. In *Proceedings of ECML*, pages 606–613.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

Thomas G. Dietterich, Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. 2008. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23.

Roberto Esposito and Daniele P. Radicioni. 2009. CARPEDIEM: Optimizing the Viterbi algorithm and applications to supervised sequential learning. *Jorunal of Machine Learning Research*, 10:1851–1880.

Pedro F. Felzenszwalb, Daniel P. Huttenlocher, and Jon M. Kleinberg. 2003. Fast algorithms for large-state-space HMMs with applications to Web usage analysis. In *Proceedings of NIPS*, pages 409–416.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Efficient inference of CRFs for large-scale natural language data. In *Proceedings of ACL-IJCNLP Short Papers*, pages 281–284.

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

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: Trading structure for features. In *Proceedings of ICML*, pages 592–599.

Yury Lifshits, Shay Mozes, Oren Weimann, and Michal Ziv-Ukelson. 2007. Speeding up HMM decoding and training by exploiting sequence repetitions. *Computational Pattern Matching*, pages 4–15.

Dekang Lin and Xiaoyun Wu. 2009. Phrae clustering for discriminative training. In *Proceedings of ACL-IJCNLP*, pages 1030–1038.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of IJCAI*, pages 1671–1676.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of EMNLP*, pages 155–163.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of The IEEE*, pages 257–286.

Sunita Sarawagi and Willian W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 134–141.

Sajid M. Siddiqi and Andrew W. Moore. 2005. Fast inference and learning in large-state-space HMMs. In *Proceedings of ICML*, pages 800–807.

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

Ben Tasker, Carlos Guestrin, and Daphe Koller. 2003. Max-margin Markov networks. In *Proceedings of NIPS*, pages 25–32.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of HLT/EMNLP*, pages 467–474.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymeptotically optimum decoding algorithm. *IEEE Transactios on Information Theory*, 13(2):260–267.

Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons

Sujith Ravi[1] **Jason Baldridge**[2] **Kevin Knight**[1]

[1]University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

[2]Department of Linguistics
The University of Texas at Austin
Austin, Texas 78712
jbaldrid@mail.utexas.edu

Abstract

We combine two complementary ideas for learning supertaggers from highly ambiguous lexicons: grammar-informed tag transitions and models minimized via integer programming. Each strategy on its own greatly improves performance over basic expectation-maximization training with a bitag Hidden Markov Model, which we show on the CCGbank and CCG-TUT corpora. The strategies provide further error reductions when combined. We describe a new two-stage integer programming strategy that efficiently deals with the high degree of ambiguity on these datasets while obtaining the full effect of model minimization.

1 Introduction

Creating accurate part-of-speech (POS) taggers using a tag dictionary and unlabeled data is an interesting task with practical applications. It has been explored at length in the literature since Merialdo (1994), though the task setting as usually defined in such experiments is somewhat artificial since the tag dictionaries are derived from tagged corpora. Nonetheless, the methods proposed apply to realistic scenarios in which one has an electronic part-of-speech tag dictionary or a hand-crafted grammar with limited coverage.

Most work has focused on POS-tagging for English using the Penn Treebank (Marcus et al., 1993), such as (Banko and Moore, 2004; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008; Goldberg et al., 2008; Ravi and Knight, 2009). This generally involves working with the standard set of 45 POS-tags employed in the Penn Treebank. The most ambiguous word has 7 different POS tags associated with it. Most methods have employed some variant of Expectation Maximization (EM) to learn parameters for a bigram or trigram Hidden Markov Model (HMM). Ravi and Knight (2009) achieved the best results thus far (92.3% word token accuracy) via a Minimum Description Length approach using an integer program (IP) that finds a minimal bigram grammar that obeys the tag dictionary constraints and covers the observed data.

A more challenging task is learning supertaggers for lexicalized grammar formalisms such as Combinatory Categorial Grammar (CCG) (Steedman, 2000). For example, CCGbank (Hockenmaier and Steedman, 2007) contains 1241 distinct supertags (lexical categories) and the most ambiguous word has 126 supertags. This provides a much more challenging starting point for the semi-supervised methods typically applied to the task. Yet, this is an important task since creating grammars and resources for CCG parsers for new domains and languages is highly labor- and knowledge-intensive. Baldridge (2008) uses grammar-informed initialization for HMM tag transitions based on the universal combinatory rules of the CCG formalism to obtain 56.1% accuracy on ambiguous word tokens, a large improvement over the 33.0% accuracy obtained with uniform initialization for tag transitions.

The strategies employed in Ravi and Knight (2009) and Baldridge (2008) are complementary. The former reduces the model size globally given a data set, while the latter biases bitag transitions toward those which are more likely based on a universal grammar without reference to any data. In this paper, we show how these strategies may be combined straightforwardly to produce improvements on the task of learning supertaggers from lexicons that have not been filtered in any way.[1] We demonstrate their cross-lingual effectiveness on CCGbank (English) and the Italian CCG-TUT

[1]See Banko and Moore (2004) for a description of how many early POS-tagging papers in fact used a number of heuristic cutoffs that greatly simplify the problem.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 495–503,
Uppsala, Sweden, 11-16 July 2010.

corpus (Bos et al., 2009). We find a consistent improved performance by using each of the methods compared to basic EM, and further improvements by using them in combination.

Applying the approach of Ravi and Knight (2009) naively to CCG supertagging is intractable due to the high level of ambiguity. We deal with this by defining a new two-stage integer programming formulation that identifies minimal grammars efficiently and effectively.

2 Data

CCGbank. CCGbank was created by semi-automatically converting the Penn Treebank to CCG derivations (Hockenmaier and Steedman, 2007). We use the standard splits of the data used in semi-supervised tagging experiments (e.g. Banko and Moore (2004)): sections 0-18 for training, 19-21 for development, and 22-24 for test.

CCG-TUT. CCG-TUT was created by semi-automatically converting dependencies in the Italian Turin University Treebank to CCG derivations (Bos et al., 2009). It is much smaller than CCGbank, with only 1837 sentences. It is split into three sections: newspaper texts (NPAPER), civil code texts (CIVIL), and European law texts from the JRC-Acquis Multilingual Parallel Corpus (JRC). For test sets, we use the first 400 sentences of NPAPER, the first 400 of CIVIL, and all of JRC. This leaves 409 and 498 sentences from NPAPER and CIVIL, respectively, for training (to acquire a lexicon and run EM). For evaluation, we use two different settings of train/test splits:

TEST 1 Evaluate on the NPAPER section of *test* using a lexicon extracted only from NPAPER section of *train*.

TEST 2 Evaluate on the entire *test* using lexicons extracted from (a) NPAPER + CIVIL, (b) NPAPER, and (c) CIVIL.

Table 1 shows statistics for supertag ambiguity in CCGbank and CCG-TUT. As a comparison, the POS word token ambiguity in CCGbank is 2.2: the corresponding value of 18.71 for supertags is indicative of the (challenging) fact that supertag ambiguity is greatest for the most frequent words.

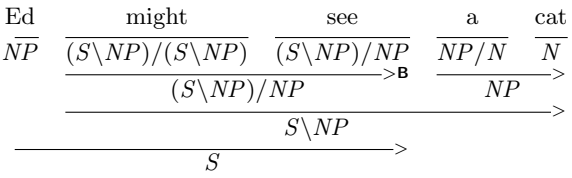
Data	Distinct	Max	Type ambig	Tok ambig
CCGbank	1241	126	1.69	18.71
CCG-TUT				
NPAPER+CIVIL	849	64	1.48	11.76
NPAPER	644	48	1.42	12.17
CIVIL	486	39	1.52	11.33

Table 1: Statistics for the *training* data used to extract lexicons for CCGbank and CCG-TUT. **Distinct**: # of distinct lexical categories; **Max**: # of categories for the most ambiguous word; **Type ambig**: per word type category ambiguity; **Tok ambig**: per word token category ambiguity.

3 Grammar informed initialization for supertagging

Part-of-speech tags are atomic labels that in and of themselves encode no internal structure. In contrast, supertags are detailed, structured labels; a universal set of grammatical rules defines how categories may combine with one another to project syntactic structure.[2] Because of this, properties of the CCG formalism itself can be used to constrain learning—prior to considering any particular language, grammar or data set. Baldridge (2008) uses this observation to create grammar-informed tag transitions for a bitag HMM supertagger based on two main properties. First, categories differ in their complexity and less complex categories tend to be used more frequently. For example, two categories for *buy* in CCGbank are (S[dcl]\NP)/NP and ((((S[b]\NP)/PP)/PP)/(S[adj]\NP))/NP; the former occurs 33 times, the latter once. Second, categories indicate the form of categories found adjacent to them; for example, the category for sentential complement verbs ((S\NP)/S) expects an NP to its left and an S to its right.

Categories combine via rules such as application and composition (see Steedman (2000) for details). Given a lexicon containing the categories for each word, these allow derivations like:



Other derivations are possible. In fact, every pair of adjacent words above may be combined directly. For example, *see* and *a* may combine through forward composition to produce the category (S\NP)/N, and *Ed*'s category may type-raise to S/(S\NP) and compose with *might*'s category.

Baldridge uses these properties to define tag

[2]Note that supertags can be lexical categories of CCG (Steedman, 2000), elementary trees of Tree-adjoining Grammar (Joshi, 1988), or types in a feature hierarchy as in Head-driven Phrase Structure Grammar (Pollard and Sag, 1994).

transition distributions that have higher likelihood for simpler categories that are able to combine. For example, for the distribution $p(t_i|t_{i-1}{=}NP)$, (S\NP)\NP is more likely than ((S\NP)/(N/N))\NP because both categories may combine with a preceding NP but the former is simpler. In turn, the latter is more likely than NP: it is more complex but can combine with the preceding NP. Finally, NP is more likely than (S/NP)/NP since neither can combine, but NP is simpler.

By starting EM with these tag transition distributions and an unfiltered lexicon (word-to-supertag dictionary), Baldridge obtains a tagging accuracy of 56.1% on ambiguous words—a large improvement over the accuracy of 33.0% obtained by starting with uniform transition distributions. We refer to a model learned from basic EM (uniformly initialized) as EM, and to a model with grammar-informed initialization as EM_{GI}.

4 Minimized models for supertagging

The idea of searching for minimized models is related to classic Minimum Description Length (MDL) (Barron et al., 1998), which seeks to select a small model that captures the most regularity in the observed data. This modeling strategy has been shown to produce good results for many natural language tasks (Goldsmith, 2001; Creutz and Lagus, 2002; Ravi and Knight, 2009). For tagging, the idea has been implemented using Bayesian models with priors that indirectly induce sparsity in the learned models (Goldwater and Griffiths, 2007); however, Ravi and Knight (2009) show a better approach is to directly minimize the model using an integer programming (IP) formulation. Here, we build on this idea for supertagging.

There are many challenges involved in using IP minimization for supertagging. The 1241 distinct supertags in the tagset result in 1.5 million tag bigram entries in the model and the dictionary contains almost 3.5 million word/tag pairs that are relevant to the test data. The set of 45 POS tags for the same data yields 2025 tag bigrams and 8910 dictionary entries. We also wish to scale our methods to larger data settings than the 24k word tokens in the test data used in the POS tagging task.

Our objective is to find the smallest supertag grammar (of tag bigram types) that explains the entire text while obeying the lexicon's constraints. However, the original IP method of Ravi and Knight (2009) is intractable for supertagging, so we propose a new two-stage method that scales to the larger tagsets and data involved.

4.1 IP method for supertagging

Our goal for supertagging is to build a minimized model with the following objective:

> $IP_{original}$: Find the smallest supertag grammar (i.e., tag bigrams) that can explain the entire text (the test word token sequence).

Using the full grammar and lexicon to perform model minimization results in a very large, difficult to solve integer program involving billions of variables and constraints. This renders the minimization objective $IP_{original}$ intractable. One way of combating this is to use a reduced grammar and lexicon as input to the integer program. We do this without further supervision by using the HMM model trained using basic EM: entries are pruned based on the tag sequence it predicts on the test data. This produces an *observed* grammar of distinct tag bigrams (G_{obs}) and lexicon of observed lexical assignments (L_{obs}). For CCGbank, G_{obs} and L_{obs} have 12,363 and 18,869 entries, respectively—far less than the millions of entries in the full grammar and lexicon.

Even though EM minimizes the model somewhat, many bad entries remain in the grammar. We prune further by supplying G_{obs} and L_{obs} as input (G, L) to the IP-minimization procedure. However, even with the EM-reduced grammar and lexicon, the IP-minimization is still very hard to solve. We thus split it into two stages. The first stage (Minimization 1) finds the smallest grammar $G_{min1} \subset G$ that explains the set of word bigram *types* observed in the data rather than the word sequence itself, and the second (Minimization 2) finds the smallest augmentation of G_{min1} that explains the full word sequence.

Minimization 1 (MIN1). We begin with a simpler minimization problem than the original one ($IP_{original}$), with the following objective:

> $IP_{\text{min 1}}$: Find the smallest set of tag bigrams $G_{min1} \subset G$, such that there is at least one tagging assignment possible for every word bigram type observed in the data.

We formulate this as an integer program, creating binary variables $gvar_i$ for every tag bigram $g_i = t_j t_k$ in G. Binary link variables connect tag bigrams with word bigrams; these are restricted

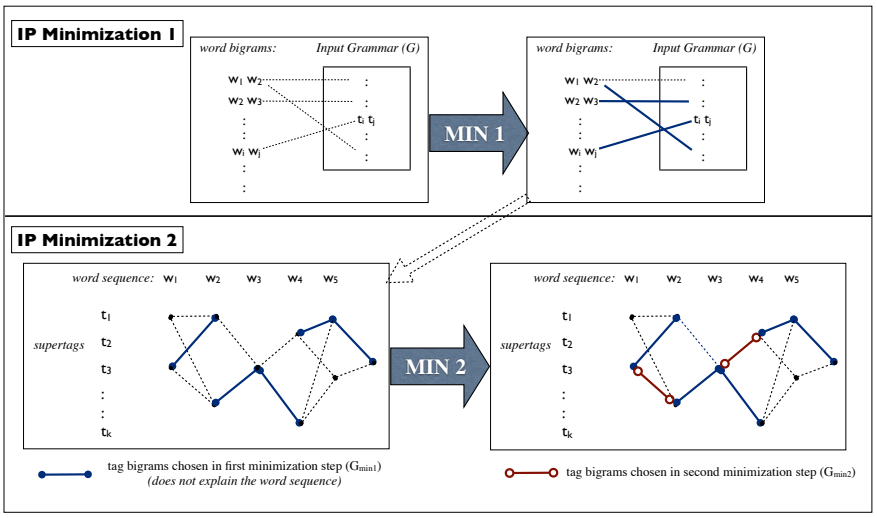


Figure 1: Two-stage IP method for selecting minimized models for supertagging.

to the set of links that respect the lexicon L provided as input, i.e., there exists a link variable $link_{jklm}$ connecting tag bigram $t_j t_k$ with word bigram $w_l w_m$ only if the word/tag pairs (w_l, t_j) and (w_m, t_k) are present in L. The entire integer programming formulation is shown Figure 2.

The IP solver[3] solves the above integer program and we extract the set of tag bigrams G_{min1} based on the activated grammar variables. For the CCGbank test data, MIN1 yields 2530 tag bigrams. However, a second stage is needed since there is no guarantee that G_{min1} can explain the test data: it contains tags for all word bigram types, but it cannot necessarily tag the full word sequence. Figure 1 illustrates this. Using only tag bigrams from MIN1 (shown in blue), there is no fully-linked tag path through the network. There are missing links between words w_2 and w_3 and between words w_3 and w_4 in the word sequence. The next stage fills in these missing links.

Minimization 2 (MIN2). This stage uses the original minimization formulation for the supertagging problem $IP_{original}$, again using an integer programming method similar to that proposed by Ravi and Knight (2009). If applied to the observed grammar G_{obs}, the resulting integer program is hard to solve.[4] However, by using the partial solution G_{min1} obtained in MIN1 the IP optimization speeds up considerably. We implement this by fixing the values of all binary grammar variables present in G_{min1} to 1 before optimization. This reduces the search space significantly, and CPLEX finishes in just a few hours. The details of this method are described below.

Minimize: $\sum_{\forall g_i \in G} gvar_i$

Subject to constraints:

1. For every word bigram $w_l w_m$, there exists at least one tagging that respects the lexicon L.

$$\sum_{\forall\ t_j \in L(w_l),\ t_k \in L(w_m)} link_{jklm} \geq 1$$

where $L(w_l)$ and $L(w_m)$ represent the set of tags seen in the lexicon for words w_l and w_m respectively.

2. The link variable assignments are constrained to respect the grammar variables chosen by the integer program.

$$link_{jklm} \leq gvar_i$$

where $gvar_i$ is the binary variable corresponding to tag bigram $t_j t_k$ in the grammar G.

Figure 2: IP formulation for Minimization 1.

We instantiate binary variables $gvar_i$ and $lvar_i$ for every tag bigram (in G) and lexicon entry (in L). We then create a network of possible taggings for the word token sequence $w_1 w_2 .... w_n$ in the corpus and assign a binary variable to each link in the network. We name these variables $link_{cjk}$, where c indicates the column of the link's source in the network, and j and k represent the link's source and destination (i.e., $link_{cjk}$ corresponds to tag bigram $t_j t_k$ in column c). Next, we formulate the integer program given in Figure 3.

Figure 1 illustrates how MIN2 augments the grammar G_{min1} (links shown in blue) with addi-

[3] We use the commercial CPLEX solver.

[4] The solver runs for days without returning a solution.

Minimize: $\sum_{\forall g_i \in G} gvar_i$

Subject to constraints:

1. Chosen link variables form a left-to-right path through the tagging network.

$$\forall_{c=1..n-2} \forall_k \sum_j link_{cjk} = \sum_j link_{(c+1)kj}$$

2. Link variable assignments should respect the chosen grammar variables.

$$\text{for every link: } link_{cjk} \leq gvar_i$$

where $gvar_i$ corresponds to tag bigram $t_j t_k$

3. Link variable assignments should respect the chosen lexicon variables.

$$\text{for every link: } link_{cjk} \leq lvar_{w_c t_j}$$

$$\text{for every link: } link_{cjk} \leq lvar_{w_{c+1} t_k}$$

where w_c is the c^{th} word in the word sequence $w_1 ... w_n$, and $lvar_{w_c t_j}$ is the binary variable corresponding to the word/tag pair w_c/t_j in the lexicon L.

4. The final solution should produce at least one complete tagging path through the network.

$$\sum_{\forall_{j,k}} link_{1jk} \geq 1$$

5. Provide minimized grammar from MIN1as partial solution to the integer program.

$$\forall_{g_i \in G_{min1}} gvar_i = 1$$

Figure 3: IP formulation for Minimization 2.

tional tag bigrams (shown in red) to form a complete tag path through the network. The minimized grammar set in the final solution G_{min2} contains only 2810 entries, significantly fewer than the original grammar G_{obs}'s 12,363 tag bigrams.

We note that the two-stage minimization procedure proposed here is not guaranteed to yield the optimal solution to our original objective $IP_{original}$. On the simpler task of unsupervised POS tagging with a dictionary, we compared our method versus *directly* solving $IP_{original}$ and found that the minimization (in terms of grammar size) achieved by our method is close to the optimal solution for the original objective and yields the same tagging accuracy far more efficiently.

Fitting the minimized model. The IP-minimization procedure gives us a minimal grammar, but does not fit the model to the data. In order to estimate probabilities for the HMM model for supertagging, we use the EM algorithm but with certain restrictions. We build the transition model using only entries from the minimized grammar set G_{min2}, and instantiate an emission model using the word/tag pairs seen in L (provided as input to the minimization procedure). All the parameters in the HMM model are initialized with uniform probabilities, and we run EM for 40 iterations. The trained model is used to find the Viterbi tag sequence for the corpus. We refer to this model (where the EM output (G_{obs}, L_{obs}) was provided to the IP-minimization as initial input) as **EM+IP**.

Bootstrapped minimization. The quality of the observed grammar and lexicon improves considerably at the end of a single EM+IP run. Ravi and Knight (2009) exploited this to iteratively improve their POS tag model: since the first minimization procedure is seeded with a noisy grammar and tag dictionary, iterating the IP procedure with progressively better grammars further improves the model. We do likewise, bootstrapping a new EM+IP run using as input, the observed grammar G_{obs} and lexicon L_{obs} from the last tagging output of the previous iteration. We run this until the chosen grammar set G_{min2} does not change.[5]

4.2 Minimization with grammar-informed initialization

There are two complementary ways to use grammar-informed initialization with the IP-minimization approach: (1) using EM_{GI} output as the starting grammar/lexicon and (2) using the tag transitions directly in the IP objective function. The first takes advantage of the earlier observation that the quality of the grammar and lexicon provided as initial input to the minimization procedure can affect the quality of the final supertagging output. For the second, we modify the objective function used in the two IP-minimization steps to be:

$$\text{Minimize: } \sum_{\forall g_i \in G} w_i \cdot gvar_i \qquad (1)$$

where, G is the set of tag bigrams provided as input to IP, $gvar_i$ is a binary variable in the integer program corresponding to tag bigram $(t_{i-1}, t_i) \in G$, and w_i is negative logarithm of $p_{gii}(t_i|t_{i-1})$ as given by Baldridge (2008).[6] All other parts of

[5] In our experiments, we run three bootstrap iterations.

[6] Other numeric weights associated with the tag bigrams could be considered, such as 0/1 for uncombin-

the integer program including the constraints remain unchanged, and, we acquire a final tagger in the same manner as described in the previous section. In this way, we combine the minimization and GI strategies into a single objective function that finds a minimal grammar set while keeping the more likely tag bigrams in the chosen solution. **EM$_{GI}$+IP$_{GI}$** is used to refer to the method that uses GI information in both ways: EM$_{GI}$ output as the starting grammar/lexicon *and* GI weights in the IP-minimization objective.

5 Experiments

We compare the four strategies described in Sections 3 and 4, summarized below:

EM HMM uniformly initialized, EM training.

EM+IP IP minimization using initial grammar provided by EM.

EM$_{GI}$ HMM with grammar-informed initialization, EM training.

EM$_{GI}$+IP$_{GI}$ IP minimization using initial grammar/lexicon provided by EM$_{GI}$ and additional grammar-informed IP objective.

For EM+IP and EM$_{GI}$+IP$_{GI}$, the minimization and EM training processes are iterated until the resulting grammar and lexicon remain unchanged. Forty EM iterations are used for all cases.

We also include a baseline which randomly chooses a tag from those associated with each word in the lexicon, averaged over three runs.

Accuracy on ambiguous word tokens. We evaluate the performance in terms of tagging accuracy with respect to gold tags for ambiguous words in held-out test sets for English and Italian. We consider results with and without punctuation.[7]

Recall that unlike much previous work, we do not collect the lexicon (tag dictionary) from the test set: this means the model must handle unknown words and the possibility of having missing lexical entries for covering the test set.

Precision and recall of grammar and lexicon. In addition to accuracy, we measure precision and recall for each model on the observed bitag grammar and observed lexicon on the test set. We calculate them as follows, for an observed grammar or lexicon X:

$$Precision = \frac{|\{X\} \cap \{Observed_{gold}\}|}{|\{X\}|}$$

$$Recall = \frac{|\{X\} \cap \{Observed_{gold}\}|}{|\{Observed_{gold}\}|}$$

This provides a measure of model performance on bitag *types* for the grammar and lexical entry *types* for the lexicon, rather than tokens.

Model	ambig	ambig -punc	all	all -punc
Random	17.9	16.2	27.4	21.9
EM	38.7	35.6	45.6	39.8
EM+IP	52.1	51.0	57.3	53.9
EM$_{GI}$	56.3	59.4	61.0	61.7
EM$_{GI}$+IP$_{GI}$	**59.6**	**62.3**	**63.8**	**64.3**

Table 2: Supertagging accuracy for CCGbank sections 22-24. Accuracies are reported for four settings—(1) *ambiguous* word tokens in the test corpus, (2) ambiguous word tokens, ignoring punctuation, (3) *all* word tokens, and (4) all word tokens except punctuation.

5.1 English CCGbank results

Accuracy on ambiguous tokens. Table 2 gives performance on the CCGbank test sections. All models are well above the random baseline, and both of the strategies individually boost performance over basic EM by a large margin. For the models using GI, accuracy ignoring punctuation is higher than for all almost entirely due to the fact that "." has the supertags "." and S, and the GI gives a preference to S since it can in fact combine with other categories, unlike "."—the effect is that nearly every sentence-final period (˜5.5k tokens) is tagged S rather than ".".

EM$_{GI}$ is more effective than EM+IP; however, it should be kept in mind that IP-minimization is a general technique that can be applied to any sequence prediction task, whereas grammar-informed initialization may be used only with tasks in which the interactions of adjacent labels may be derived from the labels themselves. Interestingly, the gap between the two approaches is greater when punctuation is ignored (51.0 vs. 59.4)—this is unsurprising because, as noted already, punctuation supertags are not actual cate-

able/combinable bigrams.

[7]The reason for this is that the "categories" for punctuation in CCGbank are for the most part not actual categories; for example, the period "." has the categories "." and "S". As such, these supertags are outside of the categorial system: their use in derivations requires phrase structure rules that are not derivable from the CCG combinatory rules.

	EM	EM+IP	EM$_{GI}$	EM$_{GI}$+IP$_{GI}$
Grammar				
Precision	7.5	32.9	52.6	68.1
Recall	26.9	13.2	34.0	19.8
Lexicon				
Precision	58.4	63.0	78.0	80.6
Recall	50.9	56.0	71.5	67.6

Table 3: Comparison of grammar/lexicon observed in the *model tagging* vs. *gold tagging* in terms of precision and recall measures for supertagging on CCGbank data.

gories, so EM$_{GI}$ is unable to model their distribution. Most importantly, the complementary effects of the two approaches can be seen in the improved results for EM$_{GI}$+IP$_{GI}$, which obtains about 3% better accuracy than EM$_{GI}$.

Accuracy on all tokens. Table 2 also gives performance when taking all tokens into account. The HMM when using full supervision obtains 87.6% accuracy (Baldridge, 2008),[8] so the accuracy of 63.8% achieved by EM$_{GI}$+IP$_{GI}$ nearly halves the gap between the supervised model and the 45.6% obtained by basic EM semi-supervised model.

Effect of GI information in EM and/or IP-minimization stages. We can also consider the effect of GI information in *either* EM training *or* IP-minimization to see whether it can be effectively exploited in both. The latter, EM+IP$_{GI}$, obtains 53.2/51.1 for all/no-punc—a small gain compared to EM+IP's 52.1/51.0. The former, EM$_{GI}$+IP, obtains 58.9/61.6—a much larger gain. Thus, the better starting point provided by EM$_{GI}$ has more impact than the integer program that includes GI in its objective function. However, we note that it should be possible to exploit the GI information more effectively in the integer program than we have here. Also, our best model, EM$_{GI}$+IP$_{GI}$, uses GI information in both stages to obtain our best accuracy of 59.6/62.3.

P/R for grammars and lexicons. We can obtain a more-fine grained understanding of how the models differ by considering the precision and recall values for the grammars and lexicons of the different models, given in Table 3. The basic EM model has very low precision for the grammar, indicating it proposes many unnecessary bitags; it achieves better recall because of the sheer number of bitags it proposes (12,363). EM+IP prunes that set of bitags considerably, leading to better precision at the cost of recall. EM$_{GI}$'s higher recall and precision indicate the tag transition distributions do capture general patterns of linkage between adjacent CCG categories, while EM ensures that the data filters out combinable, but unnecessary, bitags. With EM$_{GI}$+IP$_{GI}$, we again see that IP-minimization prunes even more entries, improving precision at the loss of some recall.

Similar trends are seen for precision and recall on the lexicon. IP-minimization's pruning of inappropriate taggings means more common words are not assigned highly infrequent supertags (boosting precision) while unknown words are generally assigned more sensible supertags (boosting recall). EM$_{GI}$ again focuses taggings on combinable contexts, boosting precision and recall similarly to EM+IP, but in greater measure. EM$_{GI}$+IP$_{GI}$ then prunes some of the spurious entries, boosting precision at some loss of recall.

Tag frequencies predicted on the test set. Table 4 compares gold tags to tags generated by all four methods for the frequent and highly ambiguous words *the* and *in*. Basic EM wanders far away from the gold assignments; it has little guidance in the very large search space available to it. IP-minimization identifies a smaller set of tags that better matches the gold tags; this emerges because other determiners and prepositions evoke similar, but not identical, supertags, and the grammar minimization pushes (but does not force) them to rely on the same supertags wherever possible. However, the proportions are incorrect; for example, the tag assigned most frequently to *in* is ((S\NP)\(S\NP))/NP though (NP\NP)/NP is more frequent in the test set. EM$_{GI}$'s tags correct that balance and find better proportions, but also some less common categories, such as (((N/N)\(N/N))\((N/N)\(N/N)))/N, sneak in because they combine with frequent categories like N/N and N. Bringing the two strategies together with EM$_{GI}$+IP$_{GI}$ filters out the unwanted categories while getting better overall proportions.

5.2 Italian CCG-TUT results

To demonstrate that both methods and their combination are language independent, we apply them to the Italian CCG-TUT corpus. We wanted to evaluate performance out-of-the-box because

[8] A state-of-the-art, fully-supervised maximum entropy tagger (Clark and Curran, 2007) (which also uses part-of-speech labels) obtains 91.4% on the same train/test split.

Lexicon	Gold	EM	EM+IP	EM$_{GI}$	EM$_{GI}$+IP$_{GI}$
the → (41 distinct tags in L_{train})	*(14 tags)*	*(18 tags)*	*(9 tags)*	*(25 tags)*	*(12 tags)*
NP[nb]/N	5742	0	4544	4176	4666
((S\NP)\(S\NP))/N	14	5	642	122	107
(((N/N)\(N/N))\((N/N)\(N/N)))/N	0	0	0	698	0
((S/S)/S[dcl])/(S[adj]\NP)	0	733	0	0	0
PP/N	0	1755	0	3	1
:	:	:	:	:	:
in → (76 distinct tags in L_{train})	*(35 tags)*	*(20 tags)*	*(17 tags)*	*(37 tags)*	*(14 tags)*
(NP\NP)/NP	883	0	649	708	904
((S\NP)\(S\NP))/NP	793	0	911	320	424
PP/NP	177	1	33	12	82
((S[adj]\NP)/(S[adj]\NP))/NP	0	215	0	0	0
:	:	:	:	:	:

Table 4: Comparison of tag assignments from the gold tags versus model tags obtained on the test set. The table shows tag assignments (and their counts for each method) for *the* and *in* in the CCGbank test sections. The number of *distinct* tags assigned by each method is given in parentheses. L_{train} is the lexicon obtained from sections 0-18 of CCGbank that is used as the basis for EM training.

Model	TEST 1	TEST 2 *(using lexicon from:)*		
		NPAPER+CIVIL	NPAPER	CIVIL
Random	9.6	9.7	8.4	9.6
EM	26.4	26.8	27.2	29.3
EM+IP	34.8	32.4	34.8	34.6
EM$_{GI}$	43.1	**43.9**	44.0	40.3
EM$_{GI}$+IP$_{GI}$	**45.8**	43.6	**47.5**	**40.9**

Table 5: Comparison of supertagging results for CCG-TUT. Accuracies are for ambiguous word tokens in the test corpus, ignoring punctuation.

	EM	EM+IP	EM$_{GI}$	EM$_{GI}$+IP$_{GI}$
Grammar				
Precision	23.1	26.4	44.9	46.7
Recall	18.4	15.9	24.9	22.7
Lexicon				
Precision	51.2	52.0	54.8	55.1
Recall	43.6	42.8	46.0	44.9

Table 6: Comparison of grammar/lexicon observed in the *model tagging* vs. *gold tagging* in terms of precision and recall measures for supertagging on CCG-TUT.

bootstrapping a supertagger for a new language is one of the main use scenarios we envision: in such a scenario, there is no development data for changing settings and parameters. Thus, we determined a train/test split beforehand and ran the methods exactly as we had for CCGbank.

The results, given in Table 5, demonstrate the same trends as for English: basic EM is far more accurate than random, EM+IP adds another 8-10% absolute accuracy, and EM$_{GI}$ adds an additional 8-10% again. The combination of the methods generally improves over EM$_{GI}$, except when the lexicon is extracted from NPAPER+CIVIL. Table 6 gives precision and recall for the grammars and lexicons for CCG-TUT—the values are lower than for CCGbank (in line with the lower baseline), but exhibit the same trends.

6 Conclusion

We have shown how two complementary strategies—grammar-informed tag transitions and IP-minimization—for learning of supertaggers from highly ambiguous lexicons can be straightforwardly integrated. We verify the benefits of both cross-lingually, on English and Italian data. We also provide a new two-stage integer programming setup that allows model minimization to be tractable for supertagging without sacrificing the quality of the search for minimal bitag grammars.

The experiments in this paper use large lexicons, but the methodology will be particularly useful in the context of bootstrapping from smaller ones. This brings further challenges; in particular, it will be necessary to identify novel entries consisting of seen word and seen category and to predict unseen, but valid, categories which are needed to explain the data. For this, it will be necessary to forgo the assumption that the provided lexicon is always obeyed. The methods we introduce here should help maintain good accuracy while opening up these degrees of freedom. Because the lexicon *is* the grammar in CCG, learning new word-category associations is grammar generalization and is of interest for grammar acquisition.

Finally, such lexicon refinement and generalization is directly relevant for using CCG in syntax-based machine translation models (Hassan et al., 2009). Such models are currently limited to languages for which corpora annotated with CCG derivations are available. Clark and Curran (2006) show that CCG parsers can be learned from sentences labeled with just supertags—without full derivations—with little loss in accuracy. The improvements we show here for learning supertaggers from lexicons without labeled data may be able to help create annotated resources more efficiently, or enable CCG parsers to be learned with less human-coded knowledge.

Acknowledgements

The authors would like to thank Johan Bos, Joey Frazee, Taesun Moon, the members of the UT-NLL reading group, and the anonymous reviewers. Ravi and Knight acknowledge the support of the NSF (grant IIS-0904684) for this work. Baldridge acknowledges the support of a grant from the Morris Memorial Trust Fund of the New York Community Trust.

References

J. Baldridge. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 57–64, Manchester, UK, August.

M. Banko and R. C. Moore. 2004. Part of speech tagging in context. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, page 556, Morristown, NJ, USA.

A. R. Barron, J. Rissanen, and B. Yu. 1998. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760.

J. Bos, C. Bosco, and A. Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 27–38, Milan, Italy.

S. Clark and J. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 144–151, New York City, USA, June.

S. Clark and J. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL Workshop on Morphological and Phonological Learning*, pages 21–30, Morristown, NJ, USA.

Y. Goldberg, M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of the ACL*, pages 746–754, Columbus, Ohio, June.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater and T. L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751, Prague, Czech Republic, June.

H. Hassan, K. Sima'an, and A. Way. 2009. A syntactified direct translation model with linear-time decoding. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1182–1191, Singapore, August.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

A. Joshi. 1988. Tree Adjoining Grammars. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

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

C. Pollard and I. Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore, August.

M. Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Kristina Toutanova and Mark Johnson. 2008. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 1521–1528, Cambridge, MA. MIT Press.

Practical very large scale CRFs

Thomas Lavergne
LIMSI – CNRS
lavergne@limsi.fr

Olivier Cappé
Télécom ParisTech
LTCI – CNRS
cappe@enst.fr

François Yvon
Université Paris-Sud 11
LIMSI – CNRS
yvon@limsi.fr

Abstract

Conditional Random Fields (CRFs) are a widely-used approach for supervised sequence labelling, notably due to their ability to handle large description spaces and to integrate structural dependency between labels. Even for the simple linear-chain model, taking structure into account implies a number of parameters and a computational effort that grows quadratically with the cardinality of the label set. In this paper, we address the issue of training very large CRFs, containing up to hundreds output labels and several billion features. Efficiency stems here from the sparsity induced by the use of a ℓ^1 penalty term. Based on our own implementation, we compare three recent proposals for implementing this regularization strategy. Our experiments demonstrate that very large CRFs can be trained efficiently and that very large models are able to improve the accuracy, while delivering compact parameter sets.

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006) constitute a widely-used and effective approach for supervised structure learning tasks involving the mapping between complex objects such as strings and trees. An important property of CRFs is their ability to handle large and redundant feature sets and to integrate structural dependency between output labels. However, even for simple *linear chain* CRFs, the complexity of learning and inference grows quadratically with respect to the number of output labels and so does the number of *structural features*, ie. features testing adjacent pairs of labels. Most empirical studies on CRFs thus either consider tasks with a restricted output space (typically in the order of few dozens of output labels), heuristically reduce the use of features, especially of features that test pairs of adjacent labels[1], and/or propose heuristics to simulate contextual dependencies, via extended tests on the observations (see discussions in, eg., (Punyakanok et al., 2005; Liang et al., 2008)). Limitating the feature set or the number of output labels is however frustrating for many NLP tasks, where the type and number of potentially relevant features are very large. A number of studies have tried to alleviate this problem. Pal et al. (2006) propose to use a "sparse" version of the forward-backward algorithm during training, where sparsity is enforced through beam pruning. Related ideas are discussed by Dietterich et al. (2004); by Cohn (2006), who considers "generalized" feature functions; and by Jeong et al. (2009), who use approximations to simplify the forward-backward recursions. In this paper, we show that the sparsity that is induced by ℓ^1-penalized estimation of CRFs can be used to reduce the total training time, while yielding extremely compact models. The benefits of sparsity are even greater during inference: less features need to be extracted and included in the potential functions, speeding up decoding with a lesser memory footprint. We study and compare three different ways to implement ℓ^1 penalty for CRFs that have been introduced recently: orthant-wise Quasi Newton (Andrew and Gao, 2007), stochastic gradient descent (Tsuruoka et al., 2009) and coordinate descent (Sokolovska et al., 2010), concluding that these methods have complemen-

This work was partly supported by ANR projects CroTaL (ANR-07-MDCO-003) and MGA (ANR-07-BLAN-0311-02).

[1] In CRFsuite (Okazaki, 2007), it is even impossible to jointly test a pair of labels and a test on the observation, bigrams feature are only of the form $f(y_{t-1}, y_t)$.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 504–513,
Uppsala, Sweden, 11-16 July 2010.

tary strengths and weaknesses. Based on an efficient implementation of these algorithms, we were able to train very large CRFs containing more than a hundred of output labels and up to several billion features, yielding results that are as good or better than the best reported results for two NLP benchmarks, text phonetization and part-of-speech tagging.

Our contribution is therefore twofold: firstly a detailed analysis of these three algorithms, discussing implementation, convergence and comparing the effect of various speed-ups. This comparison is made fair and reliable thanks to the reimplementation of these techniques in the same software package. Second, the experimental demonstration that using large output label sets is doable and that very large feature sets actually help improve prediction accuracy. In addition, we show how sparsity in *structured feature sets* can be used in incremental training regimes, where long-range features are progressively incorporated in the model insofar as the shorter range features have proven useful.

The rest of the paper is organized as follows: we first recall the basics of CRFs in Section 2, and discuss three ways to train CRFs with a ℓ^1 penalty in Section 3. We then detail several implementation issues that need to be addressed when dealing with massive feature sets in Section 4. Our experiments are reported in Section 5. The main conclusions of this study are drawn in Section 6.

2 Conditional Random Fields

In this section, we recall the basics of Conditional Random Fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006) and introduce the notations that will be used throughout.

2.1 Basics

CRFs are based on the following model

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\theta(\mathbf{x})} \exp\left\{\sum_{k=1}^{K} \theta_k F_k(\mathbf{x},\mathbf{y})\right\} \quad (1)$$

where $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{y} = (y_1, \ldots, y_T)$ are, respectively, the input and output sequences[2], and $F_k(\mathbf{x},\mathbf{y})$ is equal to $\sum_{t=1}^{T} f_k(y_{t-1}, y_t, x_t)$, where $\{f_k\}_{1\leq k\leq K}$ is an arbitrary set of feature functions and $\{\theta_k\}_{1\leq k\leq K}$ are the associated parameter values. We denote by Y and X, respectively, the sets in which y_t and x_t take their values. The normalization factor in (1) is defined by

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}\in Y^T} \exp\left\{\sum_{k=1}^{K} \theta_k F_k(\mathbf{x},\mathbf{y})\right\}. \quad (2)$$

The most common choice of feature functions is to use binary tests. In the sequel, we distinguish between two types of feature functions: *unigram features* $f_{y,x}$, associated with parameters $\mu_{y,x}$, and *bigram features* $f_{y',y,x}$, associated with parameters $\lambda_{y',y,x}$. These are defined as

$$f_{y,x}(y_{t-1}, y_t, x_t) = \mathbf{1}(y_t = y, x_t = x)$$
$$f_{y',y,x}(y_{t-1}, y_t, x_t) = \mathbf{1}(y_{t-1} = y', y_t = y, x_t = x)$$

where $\mathbf{1}(\text{cond.})$ is equal to 1 when the condition is verified and to 0 otherwise. In this setting, the number of parameters K is equal to $|Y|^2 \times |X|_{\text{train}}$, where $|\cdot|$ denotes the cardinal and $|X|_{\text{train}}$ refers to the number of configurations of x_t observed during training. Thus, even in moderate size applications, the number of parameters can be very large, mostly due to the introduction of sequential dependencies in the model. This also explains why it is hard to train CRFs with dependencies spanning more than two adjacent labels. Using only unigram features $\{f_{y,x}\}_{(y,x)\in Y\times X}$ results in a model equivalent to a simple bag-of-tokens position-by-position logistic regression model. On the other hand, bigram features $\{f_{y',y,x}\}_{(y,x)\in Y^2\times X}$ are helpful in modelling dependencies between successive labels. The motivations for using simultaneously both types of feature functions are evaluated experimentally in Section 5.

2.2 Parameter Estimation

Given N independent sequences $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ contain $T^{(i)}$ symbols, conditional maximum likelihood estimation is based on the minimization, with respect to θ, of the negated conditional log-likelihood of the observations

$$l(\theta) = -\sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \quad (3)$$
$$= \sum_{i=1}^{N} \left\{\log Z_\theta(\mathbf{x}^{(i)}) - \sum_{k=1}^{K} \theta_k F_k(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\right\}$$

This term is usually complemented with an additional regularization term so as to avoid overfitting

[2] Our implementation also includes a special label y_0, that is always observed and marks the beginning of a sequence.

(see Section 3.1 below). The gradient of $l(\theta)$ is

$$\frac{\partial l(\theta)}{\partial \theta_k} = \sum_{i=1}^{N} \sum_{t=1}^{T^{(i)}} \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\, f_k(y_{t-1}, y_t, x_t^{(i)}) - \sum_{i=1}^{N} \sum_{t=1}^{T^{(i)}} f_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}) \quad (4)$$

where $\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}$ denotes the conditional expectation given the observation sequence, i.e.

$$\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\, f_k(y_{t-1}, y_t, x_t^{(i)}) = \sum_{(y',y)\in Y^2} f_k(y, y', x_t)\, \mathrm{P}_\theta(y_{t-1} = y', y_t = y|\mathbf{x}) \quad (5)$$

Although $l(\theta)$ is a smooth convex function, its optimum cannot be computed in closed form, and $l(\theta)$ has to be optimized numerically. The computation of its gradient implies to repeatedly compute the conditional expectation in (5) for all input sequences $\mathbf{x}^{(i)}$ and all positions t. The standard approach for computing these expectations is inspired by the forward-backward algorithm for hidden Markov models: using the notations introduced above, the algorithm implies the computation of the forward

$$\begin{cases} \alpha_1(y) = \exp(\mu_{y,x_1} + \lambda_{y_0,y,x_1}) \\ \alpha_{t+1}(y) = \sum_{y'} \alpha_t(y') \exp(\mu_{y,x_{t+1}} + \lambda_{y',y,x_{t+1}}) \end{cases}$$

and backward recursions

$$\begin{cases} \beta_{T_i}(y) = 1 \\ \beta_t(y') = \sum_y \beta_{t+1}(y) \exp(\mu_{y,x_{t+1}} + \lambda_{y',y,x_{t+1}}), \end{cases}$$

for all indices $1 \leq t \leq T$ and all labels $y \in Y$. Then, $Z_\theta(\mathbf{x}) = \sum_y \alpha_T(y)$ and the pairwise probabilities $\mathrm{P}_\theta(y_t = y', y_{t+1} = y|\mathbf{x})$ are given by

$$\alpha_t(y') \exp(\mu_{y,x_{t+1}} + \lambda_{y',y,x_{t+1}}) \beta_{t+1}(y) / Z_\theta(\mathbf{x})$$

These recursions require a number of operations that grows quadratically with $|Y|$.

3 ℓ^1 Regularization in CRFs

3.1 Regularization

The standard approach for parameter estimation in CRFs consists in minimizing the logarithmic loss $l(\theta)$ defined by (3) with an additional ℓ^2 penalty term $\frac{\rho_2}{2}\|\theta\|_2^2$, where ρ_2 is a regularization parameter. The objective function is then a smooth convex function to be minimized over an unconstrained parameter space. Hence, any numerical optimization strategy may be used and practical solutions include limited memory BFGS (L-BFGS) (Liu and Nocedal, 1989), which is used in the popular CRF++ (Kudo, 2005) and CRFsuite (Okazaki, 2007) packages; conjugate gradient (Nocedal and Wright, 2006) and Stochastic Gradient Descent (SGD) (Bottou, 2004; Vishwanathan et al., 2006), used in CRFsgd (Bottou, 2007). The only caveat is to avoid numerical optimizers that require the full Hessian matrix (e.g., Newton's algorithm) due to the size of the parameter vector in usual applications of CRFs.

The most significant alternative to ℓ^2 regularization is to use a ℓ^1 penalty term $\rho_1\|\theta\|_1$: such regularizers are able to yield sparse parameter vectors in which many component have been zeroed (Tibshirani, 1996). Using a ℓ^1 penalty term thus implicitly performs feature selection, where ρ_1 controls the amount of regularization and the number of extracted features. In the following, we will jointly use both penalty terms, yielding the so-called elastic net penalty (Zhou and Hastie, 2005) which corresponds to the objective function

$$l(\theta) + \rho_1\|\theta\|_1 + \frac{\rho_2}{2}\|\theta\|_2^2 \quad (6)$$

The use of both penalty terms makes it possible to control the number of non zero coefficients and to avoid the numerical problems that might occur in large dimensional parameter settings (see also (Chen, 2009)). However, the introduction of a ℓ^1 penalty term makes the optimization of (6) more problematic, as the objective function is no longer differentiable in 0. Various strategies have been proposed to handle this difficulty. We will only consider here exact approaches and will not discuss heuristic strategies such as grafting (Perkins et al., 2003; Riezler and Vasserman, 2004).

3.2 Quasi Newton Methods

To deal with ℓ^1 penalties, a simple idea is that of (Kazama and Tsujii, 2003), originally introduced for maxent models. It amounts to reparameterizing θ_k as $\theta_k = \theta_k^+ - \theta_k^-$, where θ_k^+ and θ_k^- are positive. The ℓ^1 penalty thus becomes $\rho_1(\theta^+ - \theta^-)$. In this formulation, the objective function recovers its smoothness and can be optimized with conventional algorithms, subject to domain constraints. Optimization is straightforward, but the number of parameters is doubled and convergence is slow

(Andrew and Gao, 2007): the procedure lacks a mechanism for zeroing out useless parameters.

A more efficient strategy is the orthant-wise quasi-Newton (OWL-QN) algorithm introduced in (Andrew and Gao, 2007). The method is based on the observation that the ℓ^1 norm is differentiable when restricted to a set of points in which each coordinate never changes its sign (an "orthant"), and that its second derivative is then zero, meaning that the ℓ^1 penalty does not change the Hessian of the objective on each orthant. An OWL-QN update then simply consists in (i) computing the Newton update in a well-chosen orthant; (ii) performing the update, which might cause some component of the parameter vector to change sign; and (iii) projecting back the parameter value onto the initial orthant, thereby zeroing out those components. In (Gao et al., 2007), the authors show that OWL-QN is faster than the algorithm proposed by Kazama and Tsujii (2003) and can perform model selection even in very high-dimensional problems, with no loss of performance compared to the use of ℓ^2 penalty terms.

3.3 Stochastic Gradient Descent

Stochastic gradient (SGD) approaches update the parameter vector based on an crude approximation of the gradient (4), where the computation of expectations only includes a small batch of observations. SGD updates have the following form

$$\theta_k \leftarrow \theta_k + \eta \frac{\partial l(\theta)}{\partial \theta_k}, \tag{7}$$

where η is the learning rate. In (Tsuruoka et al., 2009), various ways of adapting this update to ℓ^1-penalized likelihood functions are discussed. Two effective ideas are proposed: (i) only update parameters that correspond to active features in the current observation, (ii) keep track of the cumulated penalty z_k that θ_k should have received, had the gradient been computed exactly, and use this value to "clip" the parameter value. This is implemented by patching the update (7) as follows

$$\begin{cases} \textbf{if } (\theta_k > 0)\ \theta_k \leftarrow \max(0, \theta_k - z_k) \\ \textbf{else if } (\theta_k < 0)\ \theta_k \leftarrow \min(0, \theta_k - z_k) \end{cases} \tag{8}$$

Based on a study of three NLP benchmarks, the authors of (Tsuruoka et al., 2009) claim this approach to be much faster than the orthant-wise approach and yet to yield very comparable performance, while selecting slightly larger feature sets.

3.4 Block Coordinate Descent

The coordinate descent approach of Dudík et al. (2004) and Friedman et al. (2008) uses the fact that optimizing a mono-dimensional quadratic function augmented with a ℓ^1 penalty can be performed analytically. For arbitrary functions, this idea can be adapted by considering quadratic approximations of the objective around the current value $\bar{\theta}$

$$l_{k,\bar{\theta}}(\theta_k) = \frac{\partial l(\bar{\theta})}{\partial \theta_k}(\theta_k - \bar{\theta}_k) + \frac{1}{2}\frac{\partial^2 l(\bar{\theta})}{\partial \theta_k^2}(\theta_k - \bar{\theta}_k)^2 + \rho_1|\theta_k| + \frac{\rho_2}{2}\theta_k^2 + C^{st} \tag{9}$$

The minimizer of the approximation (9) is simply

$$\theta_k = \frac{s\left\{\frac{\partial^2 l(\bar{\theta})}{\partial \theta_k^2}\bar{\theta}_k - \frac{\partial l(\bar{\theta})}{\partial \theta_k}, \rho_1\right\}}{\frac{\partial^2 l(\bar{\theta})}{\partial \theta_k^2} + \rho_2} \tag{10}$$

where s is the soft-thresholding function

$$s(z, \rho) = \begin{cases} z - \rho & \text{if } z > \rho \\ z + \rho & \text{if } z < -\rho \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Coordinate descent is ported to CRFs in (Sokolovska et al., 2010). Making this scheme practical requires a number of adaptations, including (i) approximating the second order term in (10), (ii) performing updates *in block*, where a block contains the $|Y| \times |Y + 1|$ features $\nu_{y',y,x}$ and $\lambda_{y,x}$ for a fixed test x on the observation sequence and (iii) approximating the Hessian for a block by its diagonal terms. (ii) is specially critical, as repeatedly cycling over individual features to perform the update (10) is only possible with restricted sets of features. The *block update* schemes uses the fact that all features within a block appear in the same set of sequences, which means that most of the computations needed to perform theses updates can be shared within the block. One advantage of the resulting algorithm, termed BCD in the following, is that the update of θ_k only involves carrying out the forward-backward recursions for the set of sequences that contain symbols x such that at least one $\{f_k(y', y, x)\}_{(y,y') \in Y^2}$ is non null, which can be much smaller than the whole training set.

4 Implementation Issues

Efficiently processing very-large feature and observation sets requires to pay attention to many implementation details. In this section, we present several optimizations devised to speed up training.

4.1 Sparse Forward-Backward Recursions

For all algorithms, the computation time is dominated by the evaluations of the gradient: our implementation takes advantage of the sparsity to accelerate these computations. Assume the set of bigram features $\{\lambda_{y',y,x_{t+1}}\}_{(y',y)\in Y^2}$ is sparse with only $r(x_{t+1}) \ll |Y|^2$ non null values and define the $|Y| \times |Y|$ sparse matrix

$$M_t(y', y) = \exp(\lambda_{y',y,x_t}) - 1.$$

Using M, the forward-backward recursions are

$$\alpha_t(y) = \sum_{y'} u_{t-1}(y') + \sum_{y'} u_{t-1}(y') M_t(y', y)$$
$$\beta_t(y') = \sum_{y} v_{t+1}(y) + \sum_{y} M_{t+1}(y', y) v_{t+1}(y)$$

with $u_{t-1}(y) = \exp(\mu_{y,x_t})\alpha_{t-1}(y)$ and $v_{t+1}(y) = \exp(\mu_{y,x_{t+1}})\beta_{t+1}(y)$. (Sokolovska et al., 2010) explains how computational savings can be obtained using the fact that the vector/matrix products in the recursions above only involve the sparse matrix $M_{t+1}(y', y)$. They can thus be computed with exactly $r(x_{t+1})$ multiplications instead of $|Y|^2$. The same idea can be used when the set $\{\mu_{y,x_{t+1}}\}_{y\in Y}$ of unigram features is sparse. Using this implementation, the complexity of the forward-backward procedure for $\mathbf{x}^{(i)}$ can be made proportional to the average number of active features per position, which can be much smaller than the number of potentially active features.

For BCD, forward-backward can even be made slightly faster. When computing the gradient wrt. features $\lambda_{y,x}$ and $\mu_{y',y,x}$ (for all the values of y and y') for sequence $\mathbf{x}^{(i)}$, assuming that x only occurs once in $\mathbf{x}^{(i)}$ at position t, all that is needed is $\alpha'_t(y), \forall t' \leq t$ and $\beta'_t(y), \forall t' \geq t$. $Z_\theta(\mathbf{x})$ is then recovered as $\sum_y \alpha_t(y)\beta_t(y)$. Forward-backward recursions can thus be truncated: in our experiments, this divided the computational cost by 1,8 on average.

Note finally that forward-backward is performed on a per-observation basis and is easily parallelized (see also (Mann et al., 2009) for more powerful ways to distribute the computation when dealing with very large datasets). In our implementation, it is distributed on all available cores, resulting in significant speed-ups for OWL-QN and L-BFGS; for BCD the gain is less acute, as parallelization only helps when updating the parameters for a block of features that are occur in many sequences; for SGD, with batches of size one, this parallelization policy is useless.

4.2 Scaling

Most existing implementations of CRFs, eg. CRF++ and CRFsgd perform the forward-backward recursions in the log-domain, which guarantees that numerical over/underflows are avoided no matter the length $T^{(i)}$ of the sequence. It is however very inefficient from an implementation point of view, due to the repeated calls to the $\exp()$ and $\log()$ functions. As an alternative way of avoiding numerical problems, our implementation, like crfSuite's, resorts to "scaling", a solution commonly used for HMMs. Scaling amounts to normalizing the values of α_t and β_t to one, making sure to keep track of the cumulated normalization factors so as to compute $Z_\theta(\mathbf{x})$ and the conditional expectations $\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}$. Also note that in our implementation, all the computations of $\exp(x)$ are vectorized, which provides an additional speed up of about 20%.

4.3 Optimization in Large Parameter Spaces

Processing very large feature vectors, up to billions of components, is problematic in many ways. Sparsity has been used here to speed up forward-backward, but we have made no attempt to accelerate the computation of the OWL-QN updates, which are linear in the size of the parameter vector. Of the three algorithms, BCD is the most affected by increases in the number of features, or more precisely, in the number of *features blocks*, where one block correspond to a specific test of the observation. In the worst case scenario, each block may require to visit all the training instances, yielding terrible computational wastes. In practice though, most blocks only require to process a small fraction of the training set, and the actual complexity depends on the average number of blocks per observations. Various strategies have been tried to further accelerate BCD, such as processing blocks that only visit one observation in parallel and updating simultaneously all the blocks that visit all the training instances, leading to a small speed-up on the POS-tagging task.

Working with billions of features finally requires to worry also about memory usage. In this respect, BCD is the most efficient, as it only requires to store one K-dimensional vector for the parameter itself. SGD requires two such vectors, one for the parameter and one for storing the z_k (see Eq. (8)). In comparison, OWL-QN requires much more memory, due to the internals of the update routines, which require several histories of the parameter vector and of its gradient. Typically, our implementation necessitates in the order of a dozen K-dimensional vectors. Parallelization only makes things worse, as each core will also need to maintain its own copy of the gradient.

5 Experiments

Our experiments use two standard NLP tasks, phonetization and part-of-speech tagging, chosen here to illustrate two very different situations, and to allow for comparison with results reported elsewhere in the literature. Unless otherwise mentioned, the experiments use the same protocol: 10 fold cross validation, where eight folds are used for training, one for development, and one for testing. Results are reported in terms of phoneme error rates or tag error rates *on the test set*.

Comparing run-times can be a tricky matter, especially when different software packages are involved. As discussed above, the observed run-times depend on many small implementation details. As the three algorithms share as much code as possible, we believe the comparison reported hereafter to be fair and reliable. All experiments were performed on a server with 64G of memory and two Xeon processors with 4 cores at 2.27 Ghz. For comparison, all measures of run-times include the cumulated activity of all cores and give very pessimistic estimates of the wall time, which can be up to 7 times smaller. For OWL-QN, we use 5 past values of the gradient to approximate the inverse of the Hessian matrix: increasing this value had no effect on accuracy or convergence and was detrimental to speed; for SGD, the learning rate parameter was tuned manually.

Note that we have not spent much time optimizing the values of ρ_1 and ρ_2. Based on a pilot study on Nettalk, we found that taking $\rho_1 = .5$ and ρ_2 in the order of 10^{-5} to yield nearly optimal performance, and have used these values throughout.

5.1 Tasks and Settings

5.1.1 Nettalk

Our first benchmark is the word phonetization task, using the Nettalk dictionary (Sejnowski and Rosenberg, 1987). This dataset contains approximately 20,000 English word forms, their pronunciation, plus some prosodic information (stress markers for vowels, syllabic parsing for consonants). Grapheme and phoneme strings are aligned at the character level, thanks to the use of a "null sound" in the latter string when it is shorter than the former; likewise, each prosodic mark is aligned with the corresponding letter. We have derived two test conditions from this database. The first one is standard and aims at predicting the pronunciation information only. In this setting, the set of observations (X) contains 26 graphemes, and the output label set contains $|Y| = 51$ phonemes.

The second condition aims at *jointly predicting phonemic and prosodic information*[3]. The reasons for designing this new condition are twofold: firstly, it yields a large set of composite labels ($|Y| = 114$) and makes the problem computationally challenging. Second, it allows to quantify how much the information provided by the prosodic marks help predict the phonemic labels. Both information are quite correlated, as the stress mark and the syllable openness, for instance, greatly influence the realization of some archi-phonemes.

The features used in Nettalk experiments take the form $f_{y,w}$ (unigram) and $f_{y',y,w}$ (bigram), where w is a n-gram of letters. The n-grm feature sets ($n = \{1, 3, 5, 7\}$) includes all features testing embedded windows of k letters, for all $0 \leq k \leq n$; the n-grm- setting is similar, but only includes the window of length n; in the n-grm+ setting, we add features for odd-size windows; in the n-grm++ setting, we add all sequences of letters up to size n occurring in current window. For instance, the active bigram features at position $t = 2$ in the sequence **x**='*lemma*' are as follows: the 3-grm feature set contains $f_{y,y'}$, $f_{y,y',e}$ and $f_{y',y,lem}$; only the latter appears in the 3-grm- setting. In the 3-grm+ feature set, we also have $f_{y',y,le}$ and $f_{y',y,em}$. The 3-grm++ feature set additionally includes $f_{y',y,l}$ and $f_{y',y,m}$. The number of features ranges from 360 thousands (1-grm setting) to 1.6 billion (7-grm).

[3]Given the design of the Nettalk dictionary, this experiment required to modify the original database so as to reassign prosodic marks to phonemes, rather than to letters.

Features	With		Without	
Nettalk				
3-grm	10.74%	$14.3M$	14.59%	$0.3M$
5-grm	8.48%	$132.5M$	11.54%	$2.5M$
POS tagging				
base	2.91%	$436.7M$	3.47%	$70.2M$

Table 1: Features jointly testing label pairs and the observation are useful (error rates and features counts.)

	ℓ^2	ℓ^1-sparse	ℓ^1	% zero
1-grm	84min	41min	57min	44.6%
3-grm-	65min	16min	44min	99.6%
3-grm	72min	48min	58min	19.9%

Table 2: Sparse vs standard forward-backward (training times and percentages of sparsity of M)

5.1.2 Part-of-Speech Tagging

Our second benchmark is a part-of-speech (POS) tagging task using the PennTreeBank corpus (Marcus et al., 1993), which provides us with a quite different condition. For this task, the number of labels is smaller ($|Y| = 45$) than for Nettalk, and the set of observations is much larger ($|X| = 43207$). This benchmark, which has been used in many studies, allows for direct comparisons with other published work. We thus use a standard experimental set-up, where sections 0-18 of the Wall Street Journal are used for training, sections 19-21 for development, and sections 22-24 for testing.

Features are also standard and follow the design of (Suzuki and Isozaki, 2008) and test the current words (as written and lowercased), prefixes and suffixes up to length 4, and typographical characteristics (case, etc.) of the words. Our baseline feature set also contains tests on individual and pairs of words in a window of 5 words.

5.2 Using Large Feature Sets

The first important issue is to assess the benefits of using large feature sets, notably including features testing both a bigram of labels and an observation. Table 1 compares the results obtained with and without these features for various setting (using OWL-QN to perform the optimization), suggesting that for the tasks at hand, these features are actually helping.

	ℓ^2	ℓ^1	Elastic-net
1-grm	17.81%	17.86%	17.79%
3-grm	10.62%	10.74%	10.70%
5-grm	8.50%	8.45%	8.48%

Table 3: Error rates of the three regularizers on the Nettalk task.

5.3 Speed, Sparsity, Convergence

The training speed depends of two main factors: the number of iterations needed to achieve convergence and the computational cost of one iteration. In this section, we analyze and compare the runtime efficiency of the three optimizers.

5.3.1 Convergence

As far as convergence is concerned, the two forms of regularization (ℓ^2 and ℓ^1) yield the same performance (see Table 3), and the three algorithms exhibit more or less the same behavior. They quickly reach an acceptable set of active parameters, which is often several orders of magnitude smaller than the whole parameter set (see results below in Table 4 and 5). Full convergence, reflected by a stabilization of the objective function, is however not so easily achieved. We have often observed a slow, yet steady, decrease of the log-loss, accompanied with a diminution of the number of active features as the number of iterations increases. Based on this observation, we have chosen to stop all algorithms based on their performance on an independent development set, allowing a fair comparison of the overall training time; for OWL-QN, it allowed to divide the total training time by almost 2.

It has finally often been found useful to fine tune the non-zero parameters by running a final handful of L-BFGS iterations using only a small ℓ^2 penalty; at this stage, all the other features are removed from the model. This had a small impact BCD and SGD's performance and allowed them to catch up with OWL-QN's performance.

5.3.2 Sparsity and the Forward-Backward

As explained in section 4.1, the forward-backward algorithm can be written so as to use the sparsity of the matrix $M_{y,y',x}$. To evaluate the resulting speed-up, we ran a series of experiments using Nettalk (see Table 2). In this table, the 3-grm- setting corresponds to maximum sparsity for M, and training with the sparse algorithm is three times faster than with the non-sparse version. Throwing

	Method	Iter.	# Feat.	Error	Time
OWL-QN	1-grm	63.4	4684	17.79%	11min
	7-grm	140.2	38214	8.12%	1h02min
	5-grm+	141.0	43429	7.89%	1h37min
SGD	1-grm	21.4	3540	18.21%	9min
	5-grm+	28.5	34319	8.01%	45min
BCD	1-grm	28.2	5017	18.27%	27min
	7-grm	9.2	3692	8.21%	1h22min
	5-grm+	8.7	47675	7.91%	2h18min

Table 4: Performance on Nettalk

	Method	Error	Time
SGD	5-grm	14.71% / 8.11%	55min
	5-grm+	13.91% / 7.51%	2h45min
BCD	5-grm	14.57% / 8.06%	2h46min
	7-grm	14.12% / 7.86%	3h02min
	5-grm+	13.85% / 7.47%	7h14min
	5-grm++	13.69% / 7.36%	16h03min

Table 5: Performance on Nettalk+prosody. Error is given for both joint labels and phonemic labels.

in more features has the effect of making M much more dense, mitigating the benefits of the sparse recursions. Nevertheless, even for very large feature sets, the percentage of zeros in M averages 20% to 30%, and the sparse version remains 10 to 20% faster than the non-sparse one. Note that the non-sparse version is faster with a ℓ^1 penalty term than with only the ℓ^2 term: this is because $\exp(0)$ is faster to evaluate than $\exp(x)$ when $x \neq 0$.

5.3.3 Training Speed and Test Accuracy

Table 4 displays the results achieved on the Nettalk task. The three algorithms yield very comparable accuracy results, and deliver compact models: for the 5-gram+ setting, only 50,000 out of 250 million features are selected. SGD is the fastest of the three, up to twice as fast as OWL-QN and BCD depending on the feature set. The performance it achieves are consistently slightly worst than the other optimizers, and only catch up when the parameters are fine-tuned (see above). There are not so many comparisons for Nettalk with CRFs, due to the size of the label set. Our results compare favorably with those reported in (Pal et al., 2006), where the accuracy attains 91.7% using 19075 examples for training and 934 for testing, and with those in (Jeong et al., 2009) (88.4% accuracy with 18,000 (2,000) training (test) instances). Table 5 gives the results obtained for the larger Nettalk+prosody task. Here, we only report the results obtained with SGD and BCD. For OWL-QN, the largest model we could handle was the 3-grm model, which contained 69 million features, and took 48min to train. Here again, performance steadily increase with the number of features, showing the benefits of large-scale models. We lack comparisons for this task, which seems considerably harder than the sole phonetization task, and all systems seem to plateau around 13.5% accuracy. Interestingly, simultaneously predicting the phoneme and its prosodic markers allows to improve the accuracy on the prediction of phonemes, which improves of almost a half point as compared to the best Nettalk system.

For the POS tagging task, BCD appears to be unpractically slower to train than the others approaches (SGD takes about 40min to train, OWL-QN about 1 hour) due the simultaneous increase in the sequence length and in the number of observations. As a result, one iteration of BCD typically requires to repeatedly process over and over the same sequences: on average, each sequence is visited 380 times when we use the baseline feature set. This technique should reserved for tasks where the number of blocks is small, or, as below, when memory usage is an issue.

5.4 Structured Feature Sets

In many tasks, the ambiguity of tokens can be reduced by looking up increasingly large windows of local context. This strategy however quickly runs into a combinatorial increase of the number of features. A side note of the Nettalk experiments is that when using embedded features, the active feature set tends to reflect this hierarchical organization. This means that when a feature testing a n-gram is active, in most cases, the features for all embedded k-grams are also selected.

Based on this observation, we have designed an incremental training strategy for the POS tagging task, where more specific features are progressively incorporated into the model if the corresponding less specific feature is active. This experiment used BCD, which is the most memory efficient algorithm. The first iteration only includes tests on the current word. During the second iteration, we add tests on bigram of words, on suffixes and prefixes up to length 4. After four iterations, we throw in features testing word trigrams, subject to the corresponding unigram block being active. After 6 iterations, we finally augment the

model with windows of length 5, subject to the corresponding trigram being active. After 10 iterations, the model contains about 4 billion features, out of which 400,000 are active. It achieves an error rate of 2.63% (resp. 2.78%) on the development (resp. test) data, which compares favorably with some of the best results for this task (for instance (Toutanova et al., 2003; Shen et al., 2007; Suzuki and Isozaki, 2008)).

6 Conclusion and Perspectives

In this paper, we have discussed various ways to train extremely large CRFs with a ℓ^1 penalty term and compared experimentally the results obtained, both in terms of training speed and of accuracy. The algorithms studied in this paper have complementary strength and weaknesses: OWL-QN is probably the method of choice in small or moderate size applications while BCD is most efficient when using very large feature sets combined with limited-size observation alphabets; SGD complemented with fine tuning appears to be the preferred choice in most large-scale applications. Our analysis demonstrate that training large-scale sparse models can be done efficiently and allows to improve over the performance of smaller models. The CRF package developed in the course of this study implements many algorithmic optimizations and allows to design innovative training strategies, such as the one presented in section 5.4. This package is released as open-source software and is available at `http://wapiti.limsi.fr`.

In the future, we intend to study how sparsity can be used to speed-up training in the face of more complex dependency patterns (such as higher-order CRFs or hierarchical dependency structures (Rozenknop, 2002; Finkel et al., 2008). From a performance point of view, it might also be interesting to combine the use of large-scale feature sets with other recent improvements such as the use of semi-supervised learning techniques (Suzuki and Isozaki, 2008) or variable-length dependencies (Qian et al., 2009).

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l1-regularized log-linear models. In *Proceedings of the International Conference on Machine Learning*, pages 33–40, Corvalis, Oregon.

Léon Bottou. 2004. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin.

Léon Bottou. 2007. Stochastic gradient descent (sgd) implementation. http://leon.bottou.org/projects/sgd.

Stanley Chen. 2009. Performance prediction for exponential language models. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 450–458, Boulder, Colorado, June.

Trevor Cohn. 2006. Efficient inference in large conditional random fields. In *Proceedings of the 17th European Conference on Machine Learning*, pages 606–613, Berlin, September.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. 2004. Training conditional random fields via gradient tree boosting. In *Proceedings of the International Conference on Machine Learning*, Banff, Canada.

Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. 2004. Performance guarantees for regularized maximum entropy density estimation. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th annual Conference on Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 472–486. Springer.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 959–967, Columbus, Ohio.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2008. Regularization paths for generalized linear models via coordinate descent. Technical report, Department of Statistics, Stanford University.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 824–831, Prague, Czech republic.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Efficient inference of crfs for large-scale natural language data. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 281–284, Suntec, Singapore.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 137–144.

Taku Kudo. 2005. CRF++: Yet another CRF toolkit. http://crfpp.sourceforge.net/.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Percy Liang, Hal Daumé, III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

Gideon Mann, Ryan McDonald, Mehryar Mohri, Nathan Silberman, and Dan Walker. 2009. Efficient large-scale distributed training of conditional maximum entropy models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A.Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1231–1239.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer.

Naoaki Okazaki. 2007. CRFsuite: A fast implementation of conditional random fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Chris Pal, Charles Sutton, and Andrew McCallum. 2006. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France.

Simon Perkins, Kevin Lacker, and James Theiler. 2003. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Vasin Punyakanok, Dan Roth, Wen tau Yih, and Dav Zimak. 2005. Learning and inference over constrained output. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1124–1129.

Xian Qian, Xiaoqian Jiang, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Sparse higher order conditional random fields for improved sequence labeling. In *Proceedings of the Annual International Conference on Machine Learning*, pages 849–856.

Stefan Riezler and Alexander Vasserman. 2004. Incremental feature selection and l1 regularization for relaxed maximum-entropy modeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 174–181, Barcelona, Spain, July.

Antoine Rozenknop. 2002. *Modèles syntaxiques probabilistes non-génératifs*. Ph.D. thesis, Dpt. d'informatique, École Polytechnique Fédérale de Lausanne.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce english text. *Complex Systems*, 1.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic.

Nataliya Sokolovska, Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Efficient learning of sparse conditional random fields for supervised sequence labelling. *IEEE Selected Topics in Signal Processing*.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, Cambridge, MA. The MIT Press.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of the Conference of the Association for Computational Linguistics on Human Language Technology*, pages 665–673, Columbus, Ohio.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *J.R.Statist.Soc.B*, 58(1):267–288.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 477–485, Suntec, Singapore.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark Schmidt, and Kevin Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23th International Conference on Machine Learning*, pages 969–976. ACM Press, New York, NY, USA.

Hui Zhou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *J. Royal. Stat. Soc. B.*, 67(2):301–320.

On the Computational Complexity of Dominance Links in Grammatical Formalisms

Sylvain Schmitz
LSV, ENS Cachan & CNRS, France
sylvain.schmitz@lsv.ens-cachan.fr

Abstract

Dominance links were introduced in grammars to model long distance scrambling phenomena, motivating the definition of *multiset-valued linear indexed grammars* (MLIGs) by Rambow (1994b), and inspiring quite a few recent formalisms. It turns out that MLIGs have since been rediscovered and reused in a variety of contexts, and that the complexity of their emptiness problem has become the key to several open questions in computer science. We survey complexity results and open issues on MLIGs and related formalisms, and provide new complexity bounds for some linguistically motivated restrictions.

1 Introduction

Scrambling constructions, as found in German and other SOV languages (Becker et al., 1991; Rambow, 1994a; Lichte, 2007), cause notorious difficulties to linguistic modeling in classical grammar formalisms like HPSG or TAG. A well-known illustration of this situation is given in the following two German sentences for "that Peter has repaired the fridge today" (Lichte, 2007),

dass [Peter] heute [den Kühlschrank] repariert hat
that Peter$_{nom}$ today the fridge$_{acc}$ repaired has

dass [den Kühlschrank] heute [Peter] repariert hat
that the fridge$_{acc}$ today Peter$_{nom}$ repaired has

with a flexible word order between the two complements of *repariert*, namely between the nominative *Peter* and the accusative *den Kühlschrank*.

Rambow (1994b) introduced a formalism, *unordered vector grammars with dominance links* (UVG-dls), for modeling such phenomena. These grammars are defined by vectors of context-free productions along with dominance links that should be enforced during derivations; for instance, Figure 1 shows how a flexible order between the complements of *repariert* could be expressed in an UVG-dl. Similar dominance mechanisms have been employed in various tree description formalisms (Rambow et al., 1995; Rambow et al., 2001; Candito and Kahane, 1998; Kallmeyer, 2001; Guillaume and Perrier, 2010) and TAG extensions (Becker et al., 1991; Rambow, 1994a).

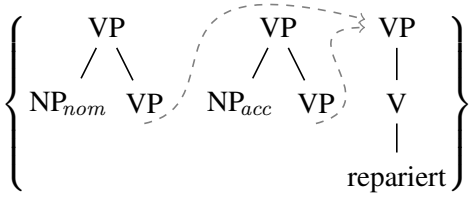


Figure 1: A vector of productions for the verb *repariert* together with its two complements.

However, the prime motivation for this survey is another grammatical formalism defined in the same article: *multiset-valued linear indexed grammars* (Rambow, 1994b, MLIGs), which can be seen as a low-level variant of UVG-dls that uses multisets to emulate unfulfilled dominance links in partial derivations. It is a natural extension of Petri nets, with broader scope than just UVG-dls; indeed, it has been independently rediscovered by de Groote et al. (2004) in the context of linear logic, and by Verma and Goubault-Larrecq (2005) in that of equational theories. Moreover, the decidability of its emptiness problem has proved to be quite challenging and is still uncertain, with several open questions depending on its resolution:

- provability in multiplicative exponential linear logic (de Groote et al., 2004),
- emptiness and membership of abstract categorial grammars (de Groote et al., 2004; Yoshinaka and Kanazawa, 2005),
- emptiness and membership of Stabler (1997)'s minimalist grammars without

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 514–524,
Uppsala, Sweden, 11-16 July 2010.

shortest move constraint (Salvati, 2010),

- satisfiability of first-order logic on data trees (Bojańczyk et al., 2009), and of course
- emptiness and membership for the various formalisms that embed UVG-dls.

Unsurprisingly in the light of their importance in different fields, several authors have started investigating the complexity of decisions problems for MLIGs (Demri et al., 2009; Lazić, 2010). We survey the current state of affairs, with a particular emphasis on two points:

1. the applicability of complexity results to UVG-dls, which is needed if we are to conclude anything on related formalisms with dominance links,
2. the effects of two linguistically motivated restrictions on such formalisms, lexicalization and boundedness/rankedness.

The latter notion is imported from Petri nets, and turns out to offer interesting new complexity trade-offs, as we prove that k-boundedness and k-rankedness are EXPTIME-complete for MLIGs, and that the emptiness and membership problems are EXPTIME-complete for k-bounded MLIGs but PTIME-complete in the k-ranked case. This also implies an EXPTIME lower bound for emptiness and membership in minimalist grammars with shortest move constraint.

We first define MLIGs formally in Section 2 and review related formalisms in Section 3. We proceed with complexity results in Section 4 before concluding in Section 5.

Notations In the following, Σ denotes a finite alphabet, Σ^* the set of finite sentences over Σ, and ε the empty string. The length of a string w is noted $|w|$, and the number of occurrence of a symbol a in w is noted $|w|_a$. A language is formalized as a subset of Σ^*. Let $\mathbb{N}^n$ denote the set of vectors of positive integers of dimension n. The i-th component of a vector $\overline{\mathsf{x}}$ in $\mathbb{N}^n$ is $\overline{\mathsf{x}}(i)$, $\overline{0}$ denotes the null vector, $\overline{1}$ the vector with 1 values, and $\overline{\mathsf{e}}_i$ the vector with 1 as its i-th component and 0 everywhere else. The ordering $\leq$ on $\mathbb{N}^n$ is the componentwise ordering: $\overline{\mathsf{x}} \leq \overline{\mathsf{y}}$ iff $\overline{\mathsf{x}}(i) \leq \overline{\mathsf{y}}(i)$ for all $0 < i \leq n$. The size of a vector refers to the size of its binary encoding: $|\overline{\mathsf{x}}| = \sum_{i=1}^{n} 1 + \max(0, \lfloor \log_2 \overline{\mathsf{x}}(i) \rfloor)$.

We refer the reader unfamiliar with complexity classes and notions such as hardness or LOGSPACE reductions to classical textbooks (e.g. Papadimitriou, 1994).

2 Multiset-Valued Linear Indexed Grammars

Definition 1 (Rambow, 1994b)**.** An n-dimensional *multiset-valued linear indexed grammar* (MLIG) is a tuple $\mathcal{G} = \langle N, \Sigma, P, (S, \overline{\mathsf{x}}_0) \rangle$ where N is a finite set of nonterminal symbols, Σ a finite alphabet disjoint from N, $V = (N \times \mathbb{N}^n) \uplus \Sigma$ the vocabulary, P a finite set of productions in $(N \times \mathbb{N}^n) \times V^*$, and $(S, \overline{\mathsf{x}}_0) \in N \times \mathbb{N}^n$ the start symbol. Productions are more easily written as

$$(A,\overline{\mathsf{x}}) \to u_0(B_1,\overline{\mathsf{x}}_1)u_1 \cdots u_m(B_m,\overline{\mathsf{x}}_m)u_{m+1} \quad (\star)$$

with each u_i in Σ^* and each $(B_i, \overline{\mathsf{x}}_i)$ in $N \times \mathbb{N}^n$.

The *derivation* relation $\Rightarrow$ over sequences in V^* is defined by

$$\delta(A,\overline{\mathsf{y}})\delta' \Rightarrow \delta u_0(B_1,\overline{\mathsf{y}}_1)u_1 \cdots u_m(B_m,\overline{\mathsf{y}}_m)u_{m+1}\delta'$$

if δ and δ' are in V^*, a production of form $(\star)$ appears in P, $\overline{\mathsf{x}} \leq \overline{\mathsf{y}}$, for each $1 \leq i \leq m$, $\overline{\mathsf{x}}_i \leq \overline{\mathsf{y}}_i$, and $\overline{\mathsf{y}} - \overline{\mathsf{x}} = \sum_{i=1}^{m} \overline{\mathsf{y}}_i - \overline{\mathsf{x}}_i$.

The *language* of a MLIG is the set of terminal strings derived from $(S, \overline{\mathsf{x}}_0)$, i.e.

$$L(\mathcal{G}) = \{w \in \Sigma^* \mid (S, \overline{\mathsf{x}}_0) \Rightarrow^* w\}$$

and we denote by $\mathcal{L}(\mathrm{MLIG})$ the class of MLIG languages.

Example 2. To illustrate this definition, and its relevance for free word order languages, consider the 3-dimensional MLIG with productions

$$(S, \overline{0}) \to \varepsilon \mid (S, \overline{1}), \qquad (S, \overline{\mathsf{e}}_1) \to a\,(S, \overline{0}),$$
$$(S, \overline{\mathsf{e}}_2) \to b\,(S, \overline{0}), \qquad (S, \overline{\mathsf{e}}_3) \to c\,(S, \overline{0})$$

and start symbol $(S, \overline{0})$. It generates the MIX language of all sentences with the same number of a, b, and c's (see Figure 2 for an example derivation):

$$L_{\text{mix}} = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}\ .$$

The *size* $|\mathcal{G}|$ of a MLIG $\mathcal{G}$ is essentially the sum of the sizes of each of its productions of form $(\star)$:

$$|\overline{\mathsf{x}}_0| + \sum_{P} \left(m + 1 + |\overline{\mathsf{x}}| + \sum_{i=1}^{m} |\overline{\mathsf{x}}_i| + \sum_{i=0}^{m+1} |u_i| \right).$$

2.1 Normal Forms

A MLIG is in *extended two form* (ETF) if all its productions are of form

terminal $(A, \overline{0}) \to a$ or $(A, \overline{0}) \to \varepsilon$, or

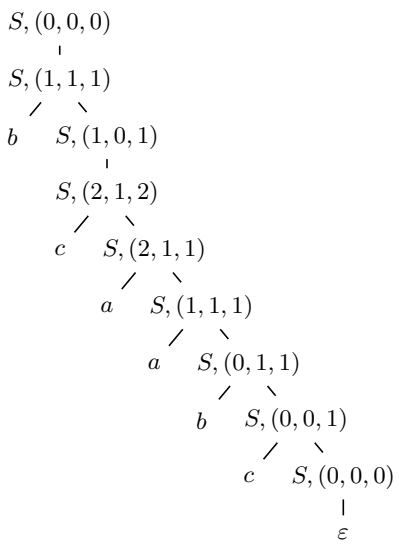


Figure 2: A derivation for $bcaabc$ in the grammar of Example 2.

nonterminal $(A,\overline{\mathsf{x}}) \rightarrow (B_1,\overline{\mathsf{x}}_1)(B_2,\overline{\mathsf{x}}_2)$ or $(A,\overline{\mathsf{x}}) \rightarrow (B_1,\overline{\mathsf{x}}_1)$,

with a in Σ, A, B_1, B_2 in N, and $\overline{\mathsf{x}}$, $\overline{\mathsf{x}}_1$, $\overline{\mathsf{x}}_2$ in $\mathbb{N}^n$. Using standard constructions, any MLIG can be put into ETF in linear time or logarithmic space.

A MLIG is in *restricted index normal form* (RINF) if the productions in P are of form $(A,\overline{0}) \rightarrow \alpha$, $(A,\overline{0}) \rightarrow (B,\overline{\mathsf{e}}_i)$, or $(A,\overline{\mathsf{e}}_i) \rightarrow (B,\overline{0})$, with A, B in N, $0 < i \leq n$, and α in $(\Sigma \cup (N \times \{\overline{0}\}))^*$. The direct translation into RINF proposed by Rambow (1994a) is exponential if we consider a binary encoding of vectors, but using techniques developed for Petri nets (Dufourd and Finkel, 1999), this blowup can be avoided:

Proposition 3. *For any MLIG, one can construct an equivalent MLIG in RINF in logarithmic space.*

2.2 Restrictions

Two restrictions on dominance links have been suggested in an attempt to reduce their complexity, sometimes in conjunction: lexicalization and k-boundedness. We provide here characterizations for them in terms of MLIGs. We can combine the two restrictions, thus defining the class of k-bounded lexicalized MLIGs.

Lexicalization Lexicalization in UVG-dls reflects the strong dependence between syntactic constructions (vectors of productions representing an extended domain of locality) and lexical anchors. We define here a restriction of MLIGs with similar complexity properties:

Definition 4. A terminal derivation $\alpha \Rightarrow^p w$ with w in Σ^* is *c-lexicalized* for some $c > 0$ if $p \leq c \cdot |w|$.[1] A MLIG is *lexicalized* if there exists c such that any terminal derivation starting from $(S,\overline{\mathsf{x}}_0)$ is c-lexicalized, and we denote by $\mathcal{L}(\mathrm{MLIG}_\ell)$ the set of lexicalized MLIG languages.

Looking at the grammar of Example 2, any terminal derivation $(S,\overline{0}) \Rightarrow^p w$ verifies $p = \frac{4 \cdot |w|}{3} + 1$, and the grammar is thus lexicalized.

Boundedness As dominance links model long-distance dependencies, bounding the number of simultaneously pending links can be motivated on competence/performance grounds (Joshi et al., 2000; Kallmeyer and Parmentier, 2008), and on complexity/expressiveness grounds (Søgaard et al., 2007; Kallmeyer and Parmentier, 2008; Chiang and Scheffler, 2008). The *shortest move constraint* (SMC) introduced by Stabler (1997) to enforce a strong form of minimality also falls into this category of restrictions.

Definition 5. A MLIG derivation $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_p$ is of *rank k* for some $k \geq 0$ if, no vector with a sum of components larger than k can appear in any α_j, i.e. for all $\overline{\mathsf{x}}$ in $\mathbb{N}^n$ such that there exist $0 \leq j \leq p$, δ, δ' in V^* and A in N with $\alpha_j = \delta(A,\overline{\mathsf{x}})\delta'$, one has $\sum_{i=1}^n \overline{\mathsf{x}}(i) \leq k$.

A MLIG is *k-ranked* (noted kr-MLIG) if any derivation starting with $\alpha_0 = (S,\overline{\mathsf{x}}_0)$ is of rank k. It is *ranked* if there exists k such that it is k-ranked.

A 0-ranked MLIG is simply a context-free grammar (CFG), and we have more generally the following:

Lemma 6. *Any n-dimensional k-ranked MLIG $\mathcal{G}$ can be transformed into an equivalent CFG $\mathcal{G}'$ in time $O(|\mathcal{G}| \cdot (n+1)^{k^3})$.*

Proof. We assume $\mathcal{G}$ to be in ETF, at the expense of a linear time factor. Each A in N is then mapped to at most $(n+1)^k$ nonterminals $(A,\overline{\mathsf{y}})$ in $N' = N \times \mathbb{N}^n$ with $\sum_{i=1}^n \overline{\mathsf{y}}(i) \leq k$. Finally, for each production $(A,\overline{\mathsf{x}}) \rightarrow (B_1,\overline{\mathsf{x}}_1)(B_2,\overline{\mathsf{x}}_2)$ of P, at most $(n+1)^{k^3}$ choices are possible for productions $(A,\overline{\mathsf{y}}) \rightarrow (B_1,\overline{\mathsf{y}}_1)(B_2,\overline{\mathsf{y}}_2)$ with $(A,\overline{\mathsf{y}})$, $(B_1,\overline{\mathsf{y}}_1)$, and $(B_2,\overline{\mathsf{y}}_2)$ in N'. □

A definition quite similar to k-rankedness can be found in the Petri net literature:

[1] This restriction is slightly stronger than that of *linearly restricted* derivations (Rambow, 1994b), but still allows to capture UVG-dl lexicalization.

Definition 7. A MLIG derivation $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_p$ is *k-bounded* for some $k \geq 0$ if, no vector with a coordinate larger than k can appear in any α_j, i.e. for all $\overline{\mathsf{x}}$ in $\mathbb{N}^n$ such that there exist $0 \leq j \leq p$, δ, δ' in V^* and A in N with $\alpha_j = \delta(A, \overline{\mathsf{x}})\delta'$, and for all $1 \leq i \leq n$, one has $\overline{\mathsf{x}}(i) \leq k$.

A MLIG is *k-bounded* (noted kb-MLIG) if any derivation starting with $\alpha_0 = (S, \overline{\mathsf{x}}_0)$ is k-bounded. It is *bounded* if there exists k such that it is k-bounded.

The SMC in minimalist grammars translates exactly into 1-boundedness of the corresponding MLIGs (Salvati, 2010).

Clearly, any k-ranked MLIG is also k-bounded, and conversely any n-dimensional k-bounded MLIG is (kn)-ranked, thus a MLIG is ranked iff it is bounded. The counterpart to Lemma 6 is:

Lemma 8. *Any n-dimensional k-bounded MLIG $\mathcal{G}$ can be transformed into an equivalent CFG $\mathcal{G}'$ in time $O(|\mathcal{G}| \cdot (k+1)^{n^2})$.*

Proof. We assume $\mathcal{G}$ to be in ETF, at the expense of a linear time factor. Each A in N is then mapped to at most $(k+1)^n$ nonterminals $(A, \overline{\mathsf{y}})$ in $N' = N \times \{0, \ldots, k\}^n$. Finally, for each production $(A, \overline{\mathsf{x}}) \to (B_1, \overline{\mathsf{x}}_1)(B_2, \overline{\mathsf{x}}_2)$ of P, each nonterminal $(A, \overline{\mathsf{y}})$ of N' with $\overline{\mathsf{x}} \leq \overline{\mathsf{y}}$, and each index $0 < i \leq n$, there are at most $k+1$ ways to split $(\overline{\mathsf{y}}(i) - \overline{\mathsf{x}}(i)) \leq k$ into $\overline{\mathsf{y}}_1(i) + \overline{\mathsf{y}}_2(i)$ and span a production $(A, \overline{\mathsf{y}}) \to (B_1, \overline{\mathsf{x}}_1 + \overline{\mathsf{y}}_1)(B_2, \overline{\mathsf{x}}_2 + \overline{\mathsf{y}}_2)$ of P'. Overall, each production is mapped to at most $(k+1)^{n^2}$ context-free productions. □

One can check that the grammar of Example 2 is not bounded (to see this, repeatedly apply production $(S, \overline{0}) \to (S, \overline{1})$), as expected since MIX is not a context-free language.

2.3 Language Properties

Let us mention a few more results pertaining to MLIG languages:

Proposition 9 (Rambow, 1994b)**.** *$\mathcal{L}(\mathrm{MLIG})$ is a substitution closed full abstract family of languages.*

Proposition 10 (Rambow, 1994b)**.** *$\mathcal{L}(\mathrm{MLIG}_\ell)$ is a subset of the context-sensitive languages.*

Natural languages are known for displaying some limited cross-serial dependencies, as witnessed in linguistic analyses, e.g. of Swiss-German (Shieber, 1985), Dutch (Kroch and Santorini, 1991), or Tagalog (Maclachlan and Rambow, 2002). This includes the copy language

$$L_{\text{copy}} = \{ww \mid w \in \{a, b\}^*\} ,$$

which does not seem to be generated by any MLIG:

Conjecture 11 (Rambow, 1994b)**.** *L_{copy} is not in $\mathcal{L}(\mathrm{MLIG})$.*

Finally, we obtain the following result as a consequence of Lemmas 6 and 8:

Corollary 12. *$\mathcal{L}(k\text{r-MLIG}) = \mathcal{L}(k\text{b-MLIG}) = \mathcal{L}(k\text{b-MLIG}_\ell)$ is the set of context-free languages.*

3 Related Formalisms

We review formalisms connected to MLIGs, starting in Section 3.1 with Petri nets and two of their extensions, which turn out to be exactly equivalent to MLIGs. We then consider various linguistic formalisms that employ dominance links (Section 3.2).

3.1 Petri Nets

Definition 13 (Petri, 1962)**.** A marked *Petri net*[2] is a tuple $\mathcal{N} = \langle \mathcal{S}, T, f, \overline{\mathsf{m}}_0 \rangle$ where $\mathcal{S}$ and T are disjoint finite sets of places and transitions, f a flow function from $(\mathcal{S} \times T) \cup (T \times \mathcal{S})$ to $\mathbb{N}$, and $\overline{\mathsf{m}}_0$ an initial marking in $\mathbb{N}^{\mathcal{S}}$. A transition $t \in T$ can be fired in a *marking* $\overline{\mathsf{m}}$ in $\mathbb{N}^{\mathcal{S}}$ if $f(p,t) \geq \overline{\mathsf{m}}(p)$ for all $p \in \mathcal{S}$, and reaches a new marking $\overline{\mathsf{m}}'$ defined by $\overline{\mathsf{m}}'(p) = \overline{\mathsf{m}}(p) - f(p,t) + f(t,p)$ for all $p \in \mathcal{S}$, written $\overline{\mathsf{m}}\ [t\rangle\ \overline{\mathsf{m}}'$. Another view is that place p holds $\overline{\mathsf{m}}(p)$ *tokens*, $f(p,t)$ of which are first removed when firing t, and then $f(t,p)$ added back. Firings are extended to sequences σ in T^* by $\overline{\mathsf{m}}\ [\varepsilon\rangle\ \overline{\mathsf{m}}$, and $\overline{\mathsf{m}}\ [\sigma t\rangle\ \overline{\mathsf{m}}'$ if there exists $\overline{\mathsf{m}}''$ with $\overline{\mathsf{m}}\ [\sigma\rangle\ \overline{\mathsf{m}}''\ [t\rangle\ \overline{\mathsf{m}}'$.

A *labeled* Petri net with reachability acceptance is endowed with a labeling homomorphism $\varphi : T^* \to \Sigma^*$ and a finite acceptance set $F \subseteq \mathbb{N}^{\mathcal{S}}$, defining the language (Peterson, 1981)

$$L(\mathcal{N}, \varphi, F) = \{\varphi(\sigma) \in \Sigma^* \mid \exists \overline{\mathsf{m}} \in F, \overline{\mathsf{m}}_0\ [\sigma\rangle\ \overline{\mathsf{m}}\} .$$

Labeled Petri nets (with acceptance set $\{\overline{0}\}$) are notational variants of *right linear* MLIGs, defined as having production in $(N \times \mathbb{N}^n) \times (\Sigma^* \cup (\Sigma^* \cdot (N \times \mathbb{N}^n)))$. This is is case of the MLIG of Example 2, which is given in Petri net form in Figure 3, where

[2] Petri nets are also equivalent to *vector addition system* (Karp and Miller, 1969, VAS) and *vector addition systems with states* (Hopcroft and Pansiot, 1979, VASS).

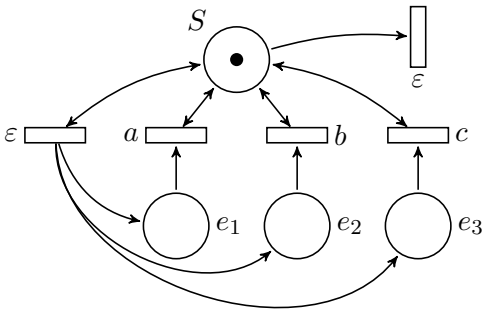


Figure 3: The labeled Petri net corresponding to the right linear MLIG of Example 2.

circles depict places (representing MLIG nonterminals and indices) with black dots for initial tokens (representing the MLIG start symbol), boxes transitions (representing MLIG productions), and arcs the flow values. For instance, production $(S,\overline{\mathsf{e}}_3) \to c\,(S,\overline{0})$ is represented by the rightmost, c-labeled transition, with $f(S,t) = f(e_3,t) = f(t,S) = 1$ and $f(e_1,t) = f(e_2,t) = f(t,e_1) = f(t,e_2) = f(t,e_3) = 0$.

Extensions The subsumption of Petri nets is not innocuous, as it allows to derive lower bounds on the computational complexity of MLIGs. Among several extensions of Petri net with some branching capacity (see e.g. Mayr, 1999; Haddad and Poitrenaud, 2007), two are of singular importance: It turns out that MLIGs in their full generality have since been independently rediscovered under the names *vector addition tree automata* (de Groote et al., 2004, VATA) and *branching VASS* (Verma and Goubault-Larrecq, 2005, BVASS).

Semilinearity Another interesting consequence of the subsumption of Petri nets by MLIGs is that the former generate some non semilinear languages, i.e. with a Parikh image which is not a semilinear subset of $\mathbb{N}^{|\Sigma|}$ (Parikh, 1966). Hopcroft and Pansiot (1979, Lemma 2.8) exhibit an example of a VASS with a non semilinear reachability set, which we translate as a 2-dimensional right linear MLIG with productions[3]

$$(S,\overline{\mathsf{e}}_2) \to (S,\overline{\mathsf{e}}_1), \qquad (S,\overline{0}) \to (A,\overline{0}) \mid (B,\overline{0}),$$
$$(A,\overline{\mathsf{e}}_1) \to (A,2\overline{\mathsf{e}}_2), \qquad (A,\overline{0}) \to a\,(S,\overline{0}),$$
$$(B,\overline{\mathsf{e}}_1) \to b\,(B,\overline{0}) \mid b, \quad (B,\overline{\mathsf{e}}_2) \to b\,(B,\overline{0}) \mid b$$

[3] Adding terminal symbols c in each production would result in a lexicalized grammar, still with a non semilinear language.

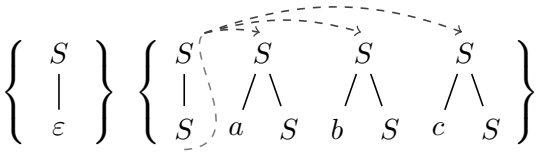


Figure 4: An UVG-dl for L_{mix}.

and $(S,\overline{\mathsf{e}}_2)$ as start symbol, that generates the non semilinear language

$$L_{\text{nsm}} = \{a^n b^m \mid 0 \le n, 0 < m \le 2^n\}\ .$$

Proposition 14 (Hopcroft and Pansiot, 1979)**.** *There exist non semilinear Petri nets languages.*

The non semilinearity of MLIGs entails that of all the grammatical formalisms mentioned next in Section 3.2; this answers in particular a conjecture by Kallmeyer (2001) about the semilinearity of V-TAGs.

3.2 Dominance Links

UVG-dl Rambow (1994b) introduced UVG-dls as a formal model for scrambling and tree description grammars.

Definition 15 (Rambow, 1994b)**.** An *unordered vector grammars with dominance links* (UVG-dl) is a tuple $\mathcal{G} = \langle N, \Sigma, W, S\rangle$ where N and Σ are disjoint finite sets of nonterminals and terminals, $V = N \cup \Sigma$ is the vocabulary, W is a set of vectors of productions with dominance links, i.e. each element of W is a pair (P, D) where each P is a multiset of productions in $N \times V^*$ and D is a relation from nonterminals in the right parts of productions in P to nonterminals in their left parts, and S in N is the start symbol.

A *terminal derivation* of w in Σ^* in an UVG-dl is a context-free derivation of form $S \overset{p_1}{\Rightarrow} \alpha_1 \overset{p_2}{\Rightarrow} \alpha_2 \cdots \alpha_{p-1} \overset{p_p}{\Rightarrow} w$ such that the control word $p_1 p_2 \cdots p_p$ is a permutation of a member of W^* and the dominance relations of W hold in the associated derivation tree. The *language* $L(\mathcal{G})$ of an UVG-dl $\mathcal{G}$ is the set of sentences w with some terminal derivation. We write $\mathcal{L}(\text{UVG-dl})$ for the class of UVG-dl languages.

An alternative semantics of derivations in UVG-dls is simply their translation into MLIGs: associate with each nonterminal in a derivation the multiset of productions it has to spawn. Figure 4 presents the two vectors of an UVG-dl for the MIX language of Example 2, with dashed arrows indicating dominance links. Observe that production

$S \to S$ in the second vector has to spawn eventually one occurrence of each $S \to aS$, $S \to bS$, and $S \to cS$, which corresponds exactly to the MLIG of Example 2.

The ease of translation from the grammar of Figure 4 into a MLIG stems from the impossibility of splitting any of its vectors (P, D) into two nonempty ones (P_1, D_1) and (P_2, D_2) while preserving the dominance relation, i.e. with $P = P_1 \uplus P_2$ and $D = D_1 \uplus D_2$. This *strictness* property can be enforced without loss of generality since we can always add to each vector (P, D) a production $S \to S$ with a dominance link to each production in P. This was performed on the second vector in Figure 4; remark that the grammar without this addition is an *unordered vector grammar* (Cremers and Mayer, 1974, UVG), and still generates L_{mix}.

Theorem 16 (Rambow, 1994b)**.** *Every MLIG can be transformed into an equivalent UVG-dl in logarithmic space, and conversely.*

Proof sketch. One can check that Rambow (1994b)'s proof of $\mathcal{L}(\text{MLIG}) \subseteq \mathcal{L}(\text{UVG-dl})$ incurs at most a quadratic blowup from a MLIG in RINF, and invoke Proposition 3. More precisely, given a MLIG in RINF, productions of form $(A,\overline{0}) \to \alpha$ with A in N and α in $(\Sigma \cup (N \times \{\overline{0}\}))^*$ form singleton vectors, and productions of form $(A,\overline{0}) \to (B,\overline{\mathsf{e}}_i)$ with A, B in N and $0 < i \leq n$ need to be paired with a production of form $(C,\overline{\mathsf{e}}_i) \to (D,\overline{0})$ for some C and D in N in order to form a vector with a dominance link between B and C.

The converse inclusion and its complexity are immediate when considering strict UVG-dls. □

The restrictions to k-ranked and k-bounded grammars find natural counterparts in strict UVG-dls by bounding the (total) number of pending dominance links in any derivation. Lexicalization has now its usual definition: for every vector $(\{p_{i,1}, \ldots, p_{i,k_i}\}, D_i)$ in W, at least one of the $p_{i,j}$ should contain at least one terminal in its right part—we have then $\mathcal{L}(\text{UVG-dl}_\ell) \subseteq \mathcal{L}(\text{MLIG}_\ell)$.

More on Dominance Links Dominance links are quite common in tree description formalisms, where they were already in use in D-theory (Marcus et al., 1983) and in quasi-tree semantics for fb-TAGs (Vijay-Shanker, 1992). In particular, D-tree substitution grammars are essentially the same as UVG-dls (Rambow et al., 2001), and quite a few other tree description formalisms subsume them (Candito and Kahane, 1998; Kallmeyer, 2001; Guillaume and Perrier, 2010). Another class of grammars are *vector TAGs* (V-TAGs), which extend TAGs and MCTAGs using dominance links (Becker et al., 1991; Rambow, 1994a; Champollion, 2007), subsuming again UVG-dls.

4 Computational Complexity

We study in this section the complexity of several decision problems on MLIGs, prominently of emptiness and membership problems, in the general (Section 4.2), k-bounded (Section 4.3), and lexicalized cases (Section 4.4). Table 1 sums up the known complexity results. Since by Theorem 16 we can translate between MLIGs and UVG-dls in logarithmic space, the complexity results on UVG-dls will be the same.

4.1 Decision Problems

Let us first review some decision problems of interest. In the following, $\mathcal{G}$ denotes a MLIG $\langle N, \Sigma, P, (S, \overline{\mathsf{x}}_0)\rangle$:

boundedness given $\langle\mathcal{G}\rangle$, is $\mathcal{G}$ bounded? As seen in Section 2.2, this is equivalent to rankedness.

***k*-boundedness** given $\langle\mathcal{G}, k\rangle$, k in $\mathbb{N}$, is $\mathcal{G}$ k-bounded? As seen in Section 2.2, this is the same as (kn)-rankedness. Here we will distinguish two cases depending on whether k is encoded in unary or binary.

coverability given $\langle\mathcal{G}, F\rangle$, $\mathcal{G}$ ε-free in ETF and F a finite subset of $N \times \mathbb{N}^n$, does there exist $\alpha = (A_1, \overline{\mathsf{y}}_1) \cdots (A_m, \overline{\mathsf{y}}_m)$ in $(N \times \mathbb{N}^n)^*$ such that $(S, \overline{\mathsf{x}}_0) \Rightarrow^* \alpha$ and for each $0 < j \leq m$ there exists $(A_j, \overline{\mathsf{x}}_j)$ in F with $\overline{\mathsf{x}}_j \leq \overline{\mathsf{y}}_j$?

reachability given $\langle\mathcal{G}, F\rangle$, $\mathcal{G}$ ε-free in ETF and F a finite subset of $N \times \mathbb{N}^n$, does there exist $\alpha = (A_1, \overline{\mathsf{y}}_1) \cdots (A_m, \overline{\mathsf{y}}_m)$ in F^* such that $(S, \overline{\mathsf{x}}_0) \Rightarrow^* \alpha$?

non emptiness given $\langle\mathcal{G}\rangle$, is $L(\mathcal{G})$ non empty?

(uniform) membership given $\langle\mathcal{G}, w\rangle$, w in Σ^*, does w belong to $L(\mathcal{G})$?

Boundedness and k-boundedness are needed in order to prove that a grammar is bounded, and to apply the smaller complexities of Section 4.3. Coverability is often considered for Petri nets, and allows to derive lower bounds on reachability. Emptiness is the most basic static

analysis one might want to perform on a grammar, and is needed for *parsing as intersection* approaches (Lang, 1994), while membership reduces to parsing. Note that we only consider uniform membership, since grammars for natural languages are typically considerably larger than input sentences, and their influence can hardly be neglected.

There are several obvious reductions between reachability, emptiness, and membership. Let $\rightarrow_{\log}$ denote LOGSPACE reductions between decision problems; we have:

Proposition 17.

$$\text{coverability} \rightarrow_{\log} \text{reachability} \quad (1)$$
$$\leftrightarrow_{\log} \text{non emptiness} \quad (2)$$
$$\leftrightarrow_{\log} \text{membership} \quad (3)$$

Proof sketch. For (1), construct a reachability instance $\langle \mathcal{G}', \{(E, \overline{0})\}\rangle$ from a coverability instance $\langle \mathcal{G}, F\rangle$ by adding to $\mathcal{G}$ a fresh nonterminal E and the productions

$$\{(A, \overline{\mathsf{x}}) \to (E, \overline{0}) \mid (A, \overline{\mathsf{x}}) \in F\}$$
$$\cup \{(E, \overline{\mathsf{e}_i}) \to (E, \overline{0}) \mid 0 < i \leq n\} .$$

For (2), from a reachability instance $\langle \mathcal{G}, F\rangle$, remove all terminal productions from $\mathcal{G}$ and add instead the productions $\{(A, \overline{\mathsf{x}}) \to \varepsilon \mid (A, \overline{\mathsf{x}}) \in F\}$; the new grammar $\mathcal{G}'$ has a non empty language iff the reachability instance was positive. Conversely, from a non emptiness instance $\langle \mathcal{G}\rangle$, put the grammar in ETF and define F to match all terminal productions, i.e. $F = \{(A, \overline{\mathsf{x}}) \mid (A, \overline{\mathsf{x}}) \to a \in P, a \in \Sigma \cup \{\varepsilon\}\}$, and then remove all terminal productions in order to obtain a reachability instance $\langle \mathcal{G}', F\rangle$.

For (3), from a non emptiness instance $\langle \mathcal{G}\rangle$, replace all terminals in $\mathcal{G}$ by ε to obtain an empty word membership instance $\langle \mathcal{G}', \varepsilon\rangle$. Conversely, from a membership instance $\langle \mathcal{G}, w\rangle$, construct the intersection grammar $\mathcal{G}'$ with $L(\mathcal{G}') = L(\mathcal{G}) \cap \{w\}$ (Bar-Hillel et al., 1961), which serves as non emptiness instance $\langle \mathcal{G}'\rangle$. □

4.2 General Case

Verma and Goubault-Larrecq (2005) were the first to prove that coverability and boundedness were decidable for BVASS, using a covering tree construction à la Karp and Miller (1969), thus of non primitive recursive complexity. Demri et al. (2009, Theorems 7, 17, and 18) recently proved tight complexity bounds for these problems, extending earlier results by Rackoff (1978) and Lipton (1976) for Petri nets.

Theorem 18 (Demri et al., 2009)**.** *Coverability and boundedness for MLIGs are* 2EXPTIME*-complete.*

Regarding reachability, emptiness, and membership, decidability is still open. A 2EXPSPACE lower bound was recently found by Lazić (2010). If a decision procedure exists, we can expect it to be quite complex, as already in the Petri net case, the complexity of the known decision procedures (Mayr, 1981; Kosaraju, 1982) is not primitive recursive (Cardoza et al., 1976, who attribute the idea to Hack).

4.3 k-Bounded and k-Ranked Cases

Since k-bounded MLIGs can be converted into CFGs (Lemma 8), emptiness and membership problems are decidable, albeit at the expense of an exponential blowup. We know from the Petri net literature that coverability and reachability problems are PSPACE-complete for k-bounded right linear MLIGs (Jones et al., 1977) by a reduction from *linear bounded automaton* (LBA) membership. We obtain the following for k-bounded MLIGs, using a similar reduction from membership in polynomially space bounded *alternating Turing machines* (Chandra et al., 1981, ATM):

Theorem 19. *Coverability and reachability for k-bounded MLIGs are* EXPTIME*-complete, even for fixed $k \geq 1$.*

The lower bound is obtained through an encoding of an instance of the membership problem for ATMs working in polynomial space into an instance of the coverability problem for 1-bounded MLIGs. The upper bound is a direct application of Lemma 8, coverability and reachability being reducible to the emptiness problem for a CFG of exponential size. Theorem 19 also shows the EXPTIME-hardness of emptiness and membership in minimalist grammars with SMC.

Corollary 20. *Let $k \geq 1$; k-boundedness for MLIGs is* EXPTIME*-complete.*

Proof. For the lower bound, consider an instance $\langle \mathcal{G}, F\rangle$ of coverability for a 1-bounded MLIG $\mathcal{G}$, which is EXPTIME-hard according to Theorem 19. Add to the MLIG $\mathcal{G}$ a fresh nonterminal E and the productions

$$\{(A, \overline{\mathsf{x}}) \to (E, \overline{\mathsf{x}}) \mid (A, \overline{\mathsf{x}}) \in F\}$$
$$\cup \{(E, \overline{0}) \to (E, \overline{\mathsf{e}_i}) \mid 0 < i \leq n\} ,$$

which make it non k-bounded iff the coverability instance was positive.

Problem	Lower bound	Upper bound
Petri net k-Boundedness	PSPACE (Jones et al., 1977)	PSPACE (Jones et al., 1977)
Petri net Boundedness	EXPSPACE (Lipton, 1976)	EXPSPACE (Rackoff, 1978)
Petri net {Emptiness, Membership}	EXPSPACE (Lipton, 1976)	Decidable, not primitive recursive (Mayr, 1981; Kosaraju, 1982)
{MLIG, MLIG$_\ell$} k-Boundedness	EXPTIME (Corollary 20)	EXPTIME (Corollary 20)
{MLIG, MLIG$_\ell$} Boundedness	2EXPTIME (Demri et al., 2009)	2EXPTIME (Demri et al., 2009)
{MLIG, MLIG$_\ell$} Emptiness MLIG Membership	2EXPSPACE (Lazić, 2010)	Not known to be decidable
{kb-MLIG, kb-MLIG$_\ell$} Emptiness kb-MLIG Membership	EXPTIME (Theorem 19)	EXPTIME (Theorem 19)
{MLIG$_\ell$, kb-MLIG$_\ell$} Membership	NPTIME (Koller and Rambow, 2007)	NPTIME (trivial)
kr-MLIG {Emptiness, Membership}	PTIME (Jones and Laaser, 1976)	PTIME (Lemma 6)

Table 1: Summary of complexity results.

For the upper bound, apply Lemma 8 with $k' = k + 1$ to construct an $O(|\mathcal{G}| \cdot 2^{n^2 \log_2(k'+1)})$-sized CFG, reduce it in polynomial time, and check whether a nonterminal $(A, \bar{\mathrm{x}})$ with $\bar{\mathrm{x}}(i) = k'$ for some $0 < i \leq n$ occurs in the reduced grammar.

Note that the choice of the encoding of k is irrelevant, as $k = 1$ is enough for the lower bound, and k only logarithmically influences the exponent for the upper bound. □

Corollary 20 also implies the EXPTIME-completeness of k-rankedness, k encoded in unary, if k can take arbitrary values. On the other hand, if k is known to be small, for instance logarithmic in the size of $\mathcal{G}$, then k-rankedness becomes polynomial by Lemma 6.

Observe finally that k-rankedness provides the only tractable class of MLIGs for uniform membership, using again Lemma 6 to obtain a CFG of polynomial size—actually exponential in k, but k is assumed to be fixed for this problem. An obvious lower bound is that of membership in CFGs, which is PTIME-complete (Jones and Laaser, 1976).

4.4 Lexicalized Case

Unlike the high complexity lower bounds of the previous two sections, NPTIME-hardness results for uniform membership have been proved for a number of formalisms related to MLIGs, from the commutative CFG viewpoint (Huynh, 1983; Barton, 1985; Esparza, 1995), or from more specialized models (Søgaard et al., 2007; Champollion, 2007; Koller and Rambow, 2007). We focus here on this last proof, which reduces from the *normal dominance graph configurability* problem (Althaus et al., 2003), as it allows to derive NPTIME-hardness even in highly restricted grammars.

Theorem 21 (Koller and Rambow, 2007)**.** *Uniform membership of $\langle \mathcal{G}, w \rangle$ for $\mathcal{G}$ a 1-bounded, lexicalized, UVG-dl with finite language is* NPTIME*-hard, even for $|w| = 1$.*

Proof sketch. Set S as start symbol and add a production $S \to aA$ to the sole vector of the grammar $\mathcal{G}$ constructed by Koller and Rambow (2007) from a normal dominance graph, with dominance links to all the other productions. Then $\mathcal{G}$ becomes strict, lexicalized, with finite language $\{a\}$ or $\emptyset$, and 1-bounded, such that a belongs to $L(\mathcal{G})$ iff the normal dominance graph is configurable. □

The fact that uniform membership is in NPTIME in the lexicalized case is clear, as we only need to guess nondeterministically a derivation of size linear in $|w|$ and check its correctness.

The weakness of lexicalized grammars is however that their emptiness problem is not any easier to solve! The effect of lexicalization is indeed to break the reduction from emptiness to membership in Proposition 17, but emptiness is as hard as ever, which means that static checks on the grammar might even be undecidable.

5 Conclusion

Grammatical formalisms with dominance links, introduced in particular to model scrambling phenomena in computational linguistics, have deep connections with several open questions in an unexpected variety of fields in computer science. We hope this survey to foster cross-fertilizing exchanges; for instance, is there a relation between

Conjecture 11 and the decidability of reachability in MLIGs? A similar question, whether the language L_{pal} of even 2-letters palindromes was a Petri net language, was indeed solved using the decidability of reachability in Petri nets (Jantzen, 1979), and shown to be strongly related to the latter (Lambert, 1992).

A conclusion with a more immediate linguistic value is that MLIGs and UVG-dls hardly qualify as formalisms for *mildly context-sensitive languages*, claimed by Joshi (1985) to be adequate for modeling natural languages, and "roughly" defined as the extensions of context-free languages that display

1. support for *limited cross-serial dependencies*: seems doubtful, see Conjecture 11,
2. constant growth, a requisite nowadays replaced by *semilinearity*: does not hold, as seen with Proposition 14, and
3. *polynomial recognition* algorithms: holds only for restricted classes of grammars, as seen in Section 4.

Nevertheless, variants such as k-ranked V-TAGs are easily seen to fulfill all the three points above.

Acknowledgements Thanks to Pierre Chambart, Stéphane Demri, and Alain Finkel for helpful discussions, and to Sylvain Salvati for pointing out the relation with minimalist grammars.

References

Ernst Althaus, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48(1):194–219.

Yehoshua Bar-Hillel, Micha Perles, and Eliahu Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172.

G. Edward Barton. 1985. The computational difficulty of ID/LP parsing. In *ACL'85*, pages 76–81. ACL Press.

Tilman Becker, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and tree adjoining grammars. In *EACL'91*, pages 21–26. ACL Press.

Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. 2009. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):1–48.

Marie-Hélène Candito and Sylvain Kahane. 1998. Defining DTG derivations to get semantic graphs. In *TAG+4*, pages 25–28.

E. Cardoza, Richard J. Lipton, and Albert R. Meyer. 1976. Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report. In *STOC'76*, pages 50–54. ACM Press.

Lucas Champollion. 2007. Lexicalized non-local MCTAG with dominance links is NP-complete. In *MOL 10*.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the ACM*, 28(1):114–133.

David Chiang and Tatjana Scheffler. 2008. Flexible composition and delayed tree-locality. In *TAG+9*.

Armin B. Cremers and Otto Mayer. 1974. On vector languages. *Journal of Computer and System Sciences*, 8(2):158–166.

Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. 2004. Vector addition tree automata. In *LICS'04*, pages 64–73. IEEE Computer Society.

Stéphane Demri, Marcin Jurdziński, Oded Lachish, and Ranko Lazić. 2009. The covering and boundedness problems for branching vector addition systems. In Ravi Kannan and K. Narayan Kumar, editors, *FSTTCS'09*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 181–192. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Catherine Dufourd and Alain Finkel. 1999. A polynomial λ-bisimilar normalization for reset Petri nets. *Theoretical Computer Science*, 222(1–2):187–194.

Javier Esparza. 1995. Petri nets, commutative context-free grammars, and basic parallel processes. In Horst Reichel, editor, *FCT'95*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232. Springer.

Bruno Guillaume and Guy Perrier. 2010. Interaction grammars. *Research on Language and Computation*. To appear.

Serge Haddad and Denis Poitrenaud. 2007. Recursive Petri nets. *Acta Informatica*, 44(7–8):463–508.

John Hopcroft and Jean-Jacques Pansiot. 1979. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159.

Dung T. Huynh. 1983. Commutative grammars: the complexity of uniform word problems. *Information and Control*, 57(1):21–39.

Matthias Jantzen. 1979. On the hierarchy of Petri net languages. *RAIRO Theoretical Informatics and Applications*, 13(1):19–30.

Neil D. Jones and William T. Laaser. 1976. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(1):105–117.

Neil D. Jones, Lawrence H. Landweber, and Y. Edmund Lien. 1977. Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4(3):277–299.

Aravind K. Joshi, Tilman Becker, and Owen Rambow. 2000. Complexity of scrambling: A new twist to the competence-performance distinction. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, chapter 6, pages 167–181. CSLI Publications.

Aravind K. Joshi. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, chapter 6, pages 206–250. Cambridge University Press.

Laura Kallmeyer and Yannick Parmentier. 2008. On the relation between multicomponent tree adjoining grammars with tree tuples (TT-MCTAG) and range concatenation grammars (RCG). In Carlos Martín-Vide, Friedrich Otto, and Henning Fernau, editors, *LATA'08*, volume 5196 of *Lecture Notes in Computer Science*, pages 263–274. Springer.

Laura Kallmeyer. 2001. Local tree description grammars. *Grammars*, 4(2):85–137.

Richard M. Karp and Raymond E. Miller. 1969. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195.

Alexander Koller and Owen Rambow. 2007. Relating dominance formalisms. In *FG'07*.

S. Rao Kosaraju. 1982. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281. ACM Press.

Anthony S. Kroch and Beatrice Santorini. 1991. The derived constituent structure of the West Germanic verb-raising construction. In Robert Freidin, editor, *Principles and Parameters in Comparative Grammar*, chapter 10, pages 269–338. MIT Press.

Jean-Luc Lambert. 1992. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104.

Bernard Lang. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494.

Ranko Lazić. 2010. The reachability problem for branching vector addition systems requires doubly-exponential space. Manuscript.

Timm Lichte. 2007. An MCTAG with tuples for coherent constructions in German. In *FG'07*.

Richard Lipton. 1976. The reachability problem requires exponential space. Technical Report 62, Yale University.

Anna Maclachlan and Owen Rambow. 2002. Cross-serial dependencies in Tagalog. In *TAG+6*, pages 100–107.

Mitchell P. Marcus, Donald Hindle, and Margaret M. Fleck. 1983. D-theory: talking about talking about trees. In *ACL'83*, pages 129–136. ACL Press.

Ernst W. Mayr. 1981. An algorithm for the general Petri net reachability problem. In *STOC'81*, pages 238–246. ACM Press.

Richard Mayr. 1999. Process rewrite systems. *Information and Computation*, 156(1–2):264–286.

Christos H. Papadimitriou. 1994. *Computational Complexity*. Addison-Wesley.

Rohit J. Parikh. 1966. On context-free languages. *Journal of the ACM*, 13(4):570–581.

James L. Peterson. 1981. *Petri Net Theory and the Modeling of Systems*. Prentice Hall.

Carl A. Petri. 1962. *Kommunikation mit Automaten*. Ph.D. thesis, University of Bonn.

Charles Rackoff. 1978. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-tree grammars. In *ACL'95*, pages 151–158. ACL Press.

Owen Rambow, David Weir, and K. Vijay-Shanker. 2001. D-tree substitution grammars. *Computational Linguistics*, 27(1):89–121.

Owen Rambow. 1994a. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania.

Owen Rambow. 1994b. Multiset-valued linear index grammars: imposing dominance constraints on derivations. In *ACL'94*, pages 263–270. ACL Press.

Sylvain Salvati. 2010. Minimalist grammars in the light of logic. Manuscript.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343.

Anders Søgaard, Timm Lichte, and Wolfgang Maier. 2007. The complexity of linguistically motivated extensions of tree-adjoining grammar. In *RANLP'07*, pages 548–553.

Edward P. Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *LACL'96*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer.

Kumar Neeraj Verma and Jean Goubault-Larrecq. 2005. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics and Theoretical Computer Science*, 7(1):217–230.

K. Vijay-Shanker. 1992. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517.

Ryo Yoshinaka and Makoto Kanazawa. 2005. The complexity and generative capacity of lexicalized abstract categorial grammars. In Philippe Blache, Edward Stabler, Joan Busquets, and Richard Moot, editors, *LACL'05*, volume 3492 of *Lecture Notes in Computer Science*, pages 330–346. Springer.

Optimal rank reduction for Linear Context-Free Rewriting Systems with Fan-Out Two

Benot Sagot
INRIA & Université Paris 7
Le Chesnay, France
benoit.sagot@inria.fr

Giorgio Satta
Department of Information Engineering
University of Padua, Italy
satta@dei.unipd.it

Abstract

Linear Context-Free Rewriting Systems (LCFRSs) are a grammar formalism capable of modeling discontinuous phrases. Many parsing applications use LCFRSs where the *fan-out* (a measure of the discontinuity of phrases) does not exceed 2. We present an efficient algorithm for optimal reduction of the length of production right-hand side in LCFRSs with fan-out at most 2. This results in asymptotical running time improvement for known parsing algorithms for this class.

1 Introduction

Linear Context-Free Rewriting Systems (LCFRSs) have been introduced by Vijay-Shanker *et al.* (1987) for modeling the syntax of natural language. The formalism extends the generative capacity of context-free grammars, still remaining far below the class of context-sensitive grammars. An important feature of LCFRSs is their ability to generate discontinuous phrases. This has been recently exploited for modeling phrase structure treebanks with discontinuous constituents (Maier and Søgaard, 2008), as well as non-projective dependency treebanks (Kuhlmann and Satta, 2009).

The maximum number f of tuple components that can be generated by an LCFRS G is called the **fan-out** of G, and the maximum number r of nonterminals in the right-hand side of a production is called the **rank** of G. As an example, context-free grammars are LCFRSs with $f = 1$ and r given by the maximum length of a production right-hand side. Tree adjoining grammars (Joshi and Levy, 1977) can also be viewed as a special kind of LCFRS with $f = 2$, since each auxiliary tree generates two strings, and with r given by the maximum number of adjunction and substitution sites in an elementary tree. Beyond tree adjoining languages, LCFRSs with $f = 2$ can also generate languages in which pair of strings derived from different nonterminals appear in so-called crossing configurations. It has recently been observed that, in this way, LCFRSs with $f = 2$ can model the vast majority of data in discontinuous phrase structure treebanks and non-projective dependency treebanks (Maier and Lichte, 2009; Kuhlmann and Satta, 2009).

Under a theoretical perspective, the parsing problem for LCFRSs with $f = 2$ is NP-complete (Satta, 1992), and in known parsing algorithms the running time is exponentially affected by the rank r of the grammar. Nonetheless, in natural language parsing applications, it is possible to achieve efficient, polynomial parsing if we succeed in reducing the rank r (number of nonterminals in the right-hand side) of individual LCFRSs' productions (Kuhlmann and Satta, 2009). This process is called production **factorization**. Production factorization is very similar to the reduction of a context-free grammar production into Chomsky normal form. However, in the LCFRS case some productions might not be reducible to $r = 2$, and the process stops at some larger value for r, which in the worst case might as well be the rank of the source production (Rambow and Satta, 1999).

Motivated by parsing efficiency, the factorization problem for LCFRSs with $f = 2$ has attracted the attention of many researchers in recent years. Most of the literature has been focusing on binarization algorithms, which attempt to find a reduction to $r = 2$ and return a failure if this is not possible. Gómez-Rodríguez *et al.* (2009) report a general binarization algorithm for LCFRS which, in the case of $f = 2$, works in time $\mathcal{O}(|p|^7)$, where $|p|$ is the size of the input production. A more efficient binarization algorithm for the case $f = 2$ is presented in (Gómez-Rodríguez and Satta, 2009), working in time $\mathcal{O}(|p|)$.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 525–533,
Uppsala, Sweden, 11-16 July 2010.

In this paper we are interested in general factorization algorithms, i.e., algorithms that find factorizations with the smallest possible rank (not necessarily $r = 2$). We present a novel technique that solves the general factorization problem in time $\mathcal{O}(|p|^2)$ for LCFRSs with $f = 2$.

Strong generative equivalence results between LCFRS and other finite copying parallel rewriting systems have been discussed in (Weir, 1992) and in (Rambow and Satta, 1999). Through these equivalence results, we can transfer the factorization techniques presented in this article to other finite copying parallel rewriting systems.

2 LCFRSs

In this section we introduce the basic notation for LCFRS and the notion of production factorization.

2.1 Definitions

Let Σ_T be a finite alphabet of terminal symbols. As usual, Σ_T^* denotes the set of all finite strings over Σ_T, including the empty string ε. For integer $k \geq 1$, $(\Sigma_T^*)^k$ denotes the set of all tuples $(w_1, \ldots, w_k)$ of strings $w_i \in \Sigma_T^*$. In what follows we are interested in functions mapping several tuples of strings in Σ_T^* into tuples of strings in Σ_T^*. Let r and f be two integers, $r \geq 0$ and $f \geq 1$. We say that a function g has **rank** r if there exist integers $f_i \geq 1$, $1 \leq i \leq r$, such that g is defined on $(\Sigma_T^*)^{f_1} \times (\Sigma_T^*)^{f_2} \times \cdots \times (\Sigma_T^*)^{f_r}$. We also say that g has **fan-out** f if the range of g is a subset of $(\Sigma_T^*)^f$. Let y_h, x_{ij}, $1 \leq h \leq f$, $1 \leq i \leq r$ and $1 \leq j \leq f_i$, be string-valued variables. A function g as above is said to be **linear regular** if it is defined by an equation of the form

$$g(\langle x_{11}, \ldots, x_{1f_1} \rangle, \ldots, \langle x_{r1}, \ldots, x_{rf_r} \rangle) = \\ = \langle y_1, \ldots, y_f \rangle, \quad (1)$$

where $\langle y_1, \ldots, y_f \rangle$ represents some grouping into f sequences of all and only the variables appearing in the left-hand side of (1) (without repetitions) along with some additional terminal symbols (with possible repetitions).

For a mathematical definition of LCFRS we refer the reader to (Weir, 1992, p. 137). Informally, in a LCFRS every nonterminal symbol A is associated with an integer $\varphi(A) \geq 1$, called its fan-out, and it generates tuples in $(\Sigma_T^*)^{\varphi(A)}$. Productions in a LCFRS have the form

$$p: \quad A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)}),$$

where $\rho(p) \geq 0$, A and B_i, $1 \leq i \leq \rho(p)$, are nonterminal symbols, and g is a linear regular function having rank $\rho(p)$ and fan-out $\varphi(A)$, defined on $(\Sigma_T^*)^{\varphi(B_1)} \times \cdots \times (\Sigma_T^*)^{\varphi(B_{\rho(p)})}$ and taking values in $(\Sigma_T^*)^{\varphi(A)}$. The basic idea underlying the rewriting relation associated with LCFRS is that production p applies to any sequence of string tuples generated by the B_i's, and provides a new string tuple in $(\Sigma_T^*)^{\varphi(A)}$ obtained through function g. We say that $\varphi(p) = \varphi(A)$ is the **fan-out** of p, and $\rho(p)$ is the **rank** of p.

Example 1 Let L be the language $L = \{a^n b^n a^m b^m a^n b^n a^m b^m \mid n, m \geq 1\}$. A LCFRS generating L is defined by means of the nonterminals S, $\varphi(S) = 1$, and A, $\varphi(A) = 2$, and the productions in figure 1. Observe that nonterminal A generates all tuples of the form $\langle a^n b^n, a^n b^n \rangle$. □

Recognition and parsing for a given LCFRS can be carried out in polynomial time on the length of the input string. This is usually done by exploiting standard dynamic programming techniques; see for instance (Seki et al., 1991).[1] However, the polynomial degree in the running time is a monotonically strictly increasing function that depends on both the rank and the fan-out of the productions in the grammar. To optimize running time, one can then recast the source grammar in such a way that the value of the above function is kept to a minimum. One way to achieve this is by factorizing the productions of a LCFRS, as we now explain.

2.2 Factorization

Consider a LCFRS production of the form $p: \quad A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)})$, where g is specified as in (1). Let also $\mathcal{C}$ be a subset of $\{B_1, B_2, \ldots, B_{\rho(p)}\}$ such that $|\mathcal{C}| \neq 0$ and $|\mathcal{C}| \neq \rho(p)$. We let $\Sigma_{\mathcal{C}}$ be the alphabet of all variables x_{ij} defined as in (1), for all values of i and j such that $B_i \in \mathcal{C}$ and $1 \leq j \leq f_i$. For each i with $1 \leq i \leq f$, we rewrite each string y_i in (1) in a form $y_i = y'_{i0} z_{i1} y'_{i1} \cdots y'_{id_{i-1}} z_{id_i} y'_{id_i}$, with $d_i \geq 0$, such that the following conditions are all met:

- each z_{ij}, $1 \leq j \leq d_i$, is a string with one or more occurrences of variables, all in $\Sigma_{\mathcal{C}}$;
- each y'_{ij}, $1 \leq j \leq d_i - 1$, is a non-empty string with no occurrences of symbols in $\Sigma_{\mathcal{C}}$;
- y'_{0j} and y'_{0d_i} are (possibly empty) strings with no occurrences of symbols in $\Sigma_{\mathcal{C}}$.

[1] In (Seki et al., 1991) a syntactic variant of LCFRS is used, called multiple context-free grammars.

$$
\begin{array}{ll}
S \rightarrow g_S(A, A), & g_S(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}x_{21}x_{12}x_{22}\rangle; \\
A \rightarrow g_A(A), & g_A(\langle x_{11}, x_{12}\rangle) = \langle ax_{11}b, ax_{12}b\rangle; \\
A \rightarrow g'_A(), & g'_A() = \langle ab, ab\rangle.
\end{array}
$$

Figure 1: A LCFRS for language $L = \{a^n b^n a^m b^m a^n b^n a^m b^m \mid n, m \geq 1\}$.

Let $c = |\mathcal{C}|$ and $\overline{c} = \rho(p) - |\mathcal{C}|$. Assume that $\mathcal{C} = \{B_{h_1}, \ldots, B_{h_c}\}$, and $\{B_1, \ldots, B_{\rho(p)}\} - \mathcal{C} = \{B_{h'_1}, \ldots, B_{h'_{\overline{c}}}\}$. We introduce a fresh nonterminal C with $\varphi(C) = \sum_{i=1}^{f} d_i$ and replace production p in our grammar by means of the two new productions $p_1 : C \rightarrow g_1(B_{h_1}, \ldots, B_{h_c})$ and $p_2 : A \rightarrow g_2(C, B_{h'_1}, \ldots, B_{h'_{\overline{c}}})$. Functions g_1 and g_2 are defined as:

$$
\begin{array}{l}
g_1(\langle x_{h_1 1}, \ldots, x_{h_1 f_{h_1}}\rangle, \ldots, \langle x_{h_c 1}, \ldots, x_{h_c f_{h_c}}\rangle) \\
\quad = \langle z_{11}, \cdots, z_{1d_1}, z_{21}, \cdots, z_{fd_f}\rangle; \\
g_2(\langle x_{h'_1 1}, \ldots, x_{h'_1 f_{h'_1}}\rangle, \ldots, \langle x_{h'_{\overline{c}} 1}, \ldots, x_{h'_{\overline{c}} f_{h'_{\overline{c}}}}\rangle) \\
\quad = \langle y'_{10}, \ldots, y'_{1d_1}, y'_{20}, \ldots, y'_{fd_f}\rangle.
\end{array}
$$

Note that productions p_1 and p_2 have rank strictly smaller than the source production p. Furthermore, if it is possible to choose set $\mathcal{C}$ in such a way that $\sum_{i=0}^{f} d_i \leq f$, then the fan-out of p_1 and p_2 will be no greater than the fan-out of p.

We can iterate the procedure above as many times as possible, under the condition that the fan-out of the productions does not increase.

Example 2 Let us consider the following production with rank 4:

$$
\begin{array}{l}
A \rightarrow g_S(B, C, D, E), \\
\quad g_A(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle, \langle x_{31}, x_{32}\rangle, \langle x_{41}, x_{42}\rangle) \\
\quad\quad = \langle x_{11}x_{21}x_{31}x_{41}x_{12}x_{42}, x_{22}x_{32}\rangle.
\end{array}
$$

Applyng the above procedure twice, we obtain a factorization consisting of three productions with rank 2 (variables have been renamed to reflect our conventions):

$$
\begin{array}{l}
A \rightarrow g_A(A_1, A_2), \\
\quad g_A(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) \\
\quad\quad = \langle x_{11}x_{21}x_{12}, x_{22}\rangle; \\
A_1 \rightarrow g_{A_1}(B, E), \\
\quad g_{A_1}(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}, x_{21}x_{12}x_{22}\rangle; \\
A_2 \rightarrow g_{A_2}(C, D), \\
\quad g_{A_2}(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) = \langle x_{11}x_{21}, x_{12}x_{22}\rangle.
\end{array}
$$

□

The factorization procedure above should be applied to all productions of a LCFRS with rank larger than two. This might result in an asymptotic improvement of the running time of existing dynamic programming algorithms for parsing based on LCFRS.

The factorization technique we have discussed can also be viewed as a generalization of well-known techniques for casting context-free grammars into binary forms. These are forms where no more than two nonterminal symbols are found in the right-hand side of productions of the grammar; see for instance (Harrison, 1978). One important difference is that, while production factorization into binary form is always possible in the context-free case, for LCFRS there are worst case grammars in which rank reduction is not possible at all, as shown in (Rambow and Satta, 1999).

3 A graph-based representation for LCFRS productions

Rather than factorizing LCFRS productions directly, in this article we work with a more abstract representation of productions based on graphs. From now on we focus on LCFRS whose nonterminals and productions all have fan-out smaller than or equal to 2. Consider then a production $p : A \rightarrow g(B_1, B_2, \ldots, B_{\rho(p)})$, with $\varphi(A), \varphi(B_i) \leq 2$, $1 \leq i \leq \rho(p)$, and with g defined as

$$
\begin{array}{l}
g(\langle x_{11}, \ldots, x_{1\varphi(B_1)}\rangle, \ldots \\
\quad \ldots, \langle x_{\rho(p)1}, \ldots, x_{\rho(p)\varphi(B_{\rho(p)})}\rangle) \\
\quad = \langle y_1, \ldots, y_{\varphi(A)}\rangle.
\end{array}
$$

In what follows, if $\varphi(A) = 1$ then $\langle y_1, \ldots, y_{\varphi(A)}\rangle$ should be read as $\langle y_1\rangle$ and $y_1 \cdots y_{\varphi(A)}$ should be read as y_1. The same convention applies to all other nonterminals and tuples.

We now introduce a special kind of undirected graph that is associated with a linear order defined over the set of its vertices. The **p-graph** associated with production p is a triple $(V_p, E_p, \prec_p)$ such that

- $V_p = \{x_{ij} \mid 1 \leq i \leq \rho(p),\ \varphi(B_i) = 2,\ 1 \leq j \leq \varphi(B_i)\}$ is a set of vertices;[2]

[2]Here we are overloading symbols x_{ij}. It will always be clear from the context whether x_{ij} is a string-valued variable or a vertex in a p-graph.

- $E_p = \{(x_{i1}, x_{i2}) \mid x_{i1}, x_{i2} \in V_p\}$ is a set of undirected edges;
- for $x, x' \in V_p$, $x \prec_p x'$ if $x \neq x'$ and the (unique) occurrence of x in $y_1 \cdots y_{\varphi(A)}$ precedes the (unique) occurrence of x'.

Note that in the above definition we are ignoring all string-valued variables x_{ij} associated with nonterminals B_i with $\varphi(B_i) = 1$. This is because nonterminals with fan-out one can always be treated as in the context-free grammar case, as it will be explained later.

Example 3 The p-graph associated with the LCFRS production in Example 2 is shown in Figure 2. Circled sets of edges indicate the factorization in that example. □

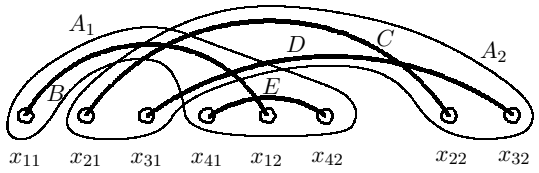


Figure 2: The p-graph associated with the LCFRS production in Example 2.

We close this section by introducing some additional notation related to p-graphs that will be used throughout this paper. Let $E \subseteq E_p$ be some set of edges. The **cover** set for E is defined as $V(E) = \{x \mid (x, x') \in E\}$ (recall that our edges are unordered pairs, so (x, x') and (x', x) denote the same edge). Conversely, let $V \subseteq V_p$ be some set of vertices. The **incident** set for V is defined as $E(V) = \{(x, x') \mid (x, x') \in E_p,\ x \in V\}$.

Assume $\varphi(p) = 2$, and let $x_1, x_2 \in V_p$. If x_1 and x_2 do not occur both in the same string y_1 or y_2, then we say that there is a **gap** between x_1 and x_2. If $x_1 \prec_p x_2$ and there is no gap between x_1 and x_2, then we write $[x_1, x_2]$ to denote the set $\{x_1, x_2\} \cup \{x \mid x \in V_p,\ x_1 \prec_p x \prec_p x_2\}$. For $x \in V_p$ we also let $[x, x] = \{x\}$. A set $[x, x']$ is called a **range**. Let r and r' be two ranges. The pair (r, r') is called a **tandem** if the following conditions are both satisfied: (i) $r \cup r'$ is not a range, and (ii) there exists some edge $(x, x') \in E_p$ with $x \in r$ and $x' \in r'$. Note that the first condition means that r and r' are disjoint sets and, for any pair of vertices $x \in r$ and $x' \in r'$, either there is a gap between x and x' or else there exists some $x_g \in V_p$ such that $x \prec_p x_g \prec_p x'$ and $x_g \notin r \cup r'$.

A set of edges $E \subseteq E_p$ is called a **bundle** with fan-out one if $V(E) = [x_1, x_2]$ for some $x_1, x_2 \in V_p$, i.e., $V(E)$ is a range. Set E is called a bundle with fan-out two if $V(E) = [x_1, x_2] \cup [x_3, x_4]$ for some $x_1, x_2, x_3, x_4 \in V_p$, and $([x_1, x_2], [x_3, x_4])$ is a tandem. Note that if E is a bundle with fan-out two with $V(E) = [x_1, x_2] \cup [x_3, x_4]$, then neither $E([x_1, x_2])$ nor $E([x_3, x_4])$ are bundles with fan-out one, since there is at least one edge incident upon a vertex in $[x_1, x_2]$ and a vertex in $[x_3, x_4]$. We also use the term bundle to denote a bundle with fan-out either one or two.

Intuitively, in a p-graph associated with a LCFRS production p, a bundle E with fan-out f and with $|E| > 1$ identifies a set of nonterminals $\mathcal{C}$ in the right-hand side of p that can be factorized into a new production. The nonterminals in $\mathcal{C}$ are then replaced in p by a fresh nonterminal C with fan-out f, as already explained. Our factorization algorithm is based on efficient methods for the detection of bundles with fan-out one and two.

4 The algorithm

In this section we provide an efficient, recursive algorithm for the decomposition of a p-graph into bundles, which corresponds to factorizing the represented LCFRS production.

4.1 Overview of the algorithm

The basic idea underlying our graph-based algorithm can be described as follows. We want to compute an optimal hierarchical decomposition of an input bundle with fan-out 1 or 2. This decomposition can be represented by a tree, in which each node N corresponds to a bundle (the root node corresponds to the input bundle) and the daughters of N represent the bundles in which N is immediately decomposed. The decomposition is optimal in so far as the maximum arity of the decomposition tree is as small as possible. As already explained above, this decomposition represents a factorization of some production p of a LCFRS, resulting in optimal rank reduction. All the internal nodes in the decomposition represent fresh nonterminals that will be created during the factorization process.

The construction of the decomposition tree is carried out recursively. For a given bundle with fan-out 1 or 2, we apply a procedure for decomposing this bundle in its immediate sub-bundles with fan-out 1 or 2, in an optimal way. Then,

we recursively apply our procedure to the obtained sub-bundles. Recursion stops when we reach bundles containing only one edge (which correspond to the nonterminals in the right-hand side of the input production). We shall prove that the result is an optimal decomposition.

The procedure for computing an optimal decomposition of a bundle F into its immediate sub-bundles, which we describe in the first part of this section, can be sketched as follows. First, we identify and temporarily remove all maximal bundles with fan-out 1 (Section 4.3). The result is a new bundle F' which is a subset of the original bundle, and has the same fan-out. Next, we identify all sub-bundles with fan-out 2 in F' (Section 4.4). We compute the optimal decomposition of F', resting on the hypothesis that there are no sub-bundles with fan-out 1. Each resulting sub-bundle is later expanded with the maximal sub-bundles with fan-out 1 that have been previously removed. This results in a "first level" decomposition of the original bundle F. We then recursively decompose all individual sub-bundles of F, including the bundles with fan-out 1 that have been later attached.

4.2 Backward and forward quantities

For a set $V \subseteq V_p$ of vertices, we write $\max(V)$ (resp. $\min(V)$) the maximum (resp. minimum) vertex in V w.r.t. the $\prec_p$ total order.

Let $r = [x_1, x_2]$ be a range. We write $r.left = x_1$ and $r.right = x_2$. The set of backward edges for r is defined as $B_r = \{(x, x') \,|\, (x, x') \in E_r,\ x \prec_p r.left,\ x' \in r\}$. The set of forward edges for r is defined symmetrically as $F_r = \{(x, x') \,|\, (x, x') \in E_r,\ x \in r,\ r.right \prec_p x'\}$. For $E \in \{B_r, F_r\}$ we also define $L(E) = \{x \,|\, (x, x') \in E,\ x \prec_p x'\}$ and $R(E) = \{x' \,|\, (x, x') \in E,\ x \prec_p x'\}$.

Let us assume $B_r \neq \emptyset$. We write $r.b.left = \min(L(B_r))$. Intuitively, $r.b.left$ is the leftmost vertex of the p-graph that is located at the left of range r and that is connected to some vertex in r through some edge. Similarly, we write $r.b.right = \max(L(B_r))$. If $B_r = \emptyset$, then we set $r.b.left = r.b.right = \bot$. Quantities $r.b.left$ and $r.b.right$ are called **backward** quantities.

We also introduce **local backward** quantities, defined as follows. We write $r.lb.left = \min(R(B_r))$. Intuitively, $r.lb.left$ is the leftmost vertex among all those vertices in r that are connected to some vertex to the left of r. Similarly, we write $r.lb.right = \max(R(B_r))$. If $B_r = \emptyset$, then we set $r.lb.left = r.lb.right = \bot$.

We define **forward** and **local forward** quantities in a symmetrical way.

The backward quantities $r.b.left$ and $r.b.right$ and the local backward quantities $r.lb.left$ and $r.lb.right$ for all ranges r in the p-graph can be computed efficiently as follows. We process ranges in increasing order of size, expanding each range r by one unit at a time by adding a new vertex at its right. Backward and local backward quantities for the expanded range can be expressed as a function of the same quantities for r. Therefore if we store our quantities for previously processed ranges, each new range can be annotated with the desired quantities in constant time. This algorithm runs in time $\mathcal{O}(n^2)$, where n is the number of vertices in V_p. This is an optimal result, since $\mathcal{O}(n^2)$ is also the size of the output.

We compute in a similar way the forward quantities $r.f.left$ and $r.f.right$ and the local forward quantities $r.lf.left$ and $r.lf.right$, this time expanding each range by one unit at its left.

4.3 Bundles with fan-out one

The detection of bundles with fan-out 1 within the p-graph can be easily performed in $\mathcal{O}(n^2)$, where n is the number of its vertices. Indeed, the incident set $E(r)$ of a range r is a bundle with fan-out one if and only if $r.b.left = r.f.left = \bot$. This immediately follows from the definitions given in Section 4.2. It is therefore possible to check all ranges the one after the other, once the backward and forward properties have been computed. These checks take constant time for each of the $\Theta(n^2)$ ranges, hence the quadratic complexity.

We now remove from F all bundles with fan-out 1 from the original bundle F. The result is the new bundle F', that has no sub-bundles with fan-out 1.

4.4 Bundles with fan-out two

Efficient detection of bundles with fan-out two in F' is considerably more challenging. A direct generalization of the technique proposed for detecting bundles with fan-out 1 would use the following property, that is also a direct corollary of the definitions in Section 4.2: the incident set $E(r \cup r')$ of a tandem (r, r') is a bundle with fan-out two if and only if all of the following conditions hold: (i) $r.b.left = r'.f.left = \bot$, (ii) $r.f.left \in r'$, $r.f.right \in r'$, (iii) $r'.b.left \in r$, $r'.b.right \in r$.

However, checking all $\mathcal{O}(n^4)$ tandems the one after the other would require time $\mathcal{O}(n^4)$. Therefore, preserving the quadratic complexity of the overall algorithm requires a more complex representation.

From now on, we assume that $V_p = \{x_1, \ldots, x_n\}$, and we write $[i, j]$ as a shorthand for the range $[x_i, x_j]$.

First, we need to compute an additional data structure that will store local backward figures in a convenient way. Let us define the **expansion table** T as follows: for a given range $r' = [i', j']$, $T(r')$ is the set of all ranges $r = [i, j]$ such that $r.lb.left = i'$ and $r.lb.right = j'$, ordered by increasing left boundary i. It turns out that the construction of such a table can be achieved in time $\mathcal{O}(n^2)$. Moreover, it is possible to compute in $\mathcal{O}(n^2)$ an auxiliary table T' that associates with r the first range r'' in $T([r.f.left, r.f.right])$ such that $r''.b.right \geq r$. Therefore, either $(r, T'(r))$ anchors a valid bundle, or there is no bundle E such that the first component of $V(E)$ is r.

We now have all the pieces to extract bundles with fan-out 2 in time $\mathcal{O}(n^2)$. We proceed as follows. For each range $r = [i, j]$:

- We first retrieve $r' = [r.f.left, r.f.right]$ in constant time.
- Then, we check in constant time whether $r'.b.left$ lies within r. If it doesn't, r is not the first part of a valid bundle with fan-out 2, and we move on to the next range r.
- Finally, for each r'' in the ordered set $T(r')$, starting with $T'(r)$, we check whether $r''.b.right$ is inside r. If it is not, we stop and move on to the next range r. If it is, we output the valid bundle (r, r'') and move on to the next element in $T(r')$. Indeed, in case of a failure, the backward edge that relates a vertex in r'' with a vertex outside r will still be included in all further elements in $T(r')$ since $T(r')$ is ordered by increasing left boundary. This step costs a constant time for each success, and a constant time for the unique failure, if any.

This algorithm spends a constant time on each range plus a constant time on each bundle with fan-out 2. We shall prove in Section 5 that there are $\mathcal{O}(n^2)$ bundles with fan-out 2. Therefore, this algorithm runs in time $\mathcal{O}(n^2)$.

Now that we have extracted all bundles, we need to extract an optimal decomposition of the input bundle F', i.e., a minimal size partition of all n elements (edges) in the input bundle such that each of these partition is a bundle (with fan-out 2, since bundles with fan-out 1 are excluded, except for the input bundle). By definition, a partition has minimal size if there is no other partition it is a refinment of.[3]

4.5 Extracting an optimal decomposition

We have constructed the set of all (fan-out 2) sub-bundles of F'. We now need to build one optimal decomposition of F' into sub-bundles. We need some more theoretical results on the properties of bundles.

Lemma 1 *Let E_1 and E_2 be two sub-bundles of F' (with fan-out 2) that have non-empty intersection, but that are not included the one in the other. Then $E_1 \cup E_2$ is a bundle (with fan-out 2).*

PROOF This lemma can be proved by considering all possible respective positions of the covers of E_1 and E_2, and discarding all situations that would lead to the existence of a fan-out 1 sub-bundle. ■

Theorem 1 *For any bundle E, either it has at least one binary decomposition, or all its decompositions are refinements of a unique optimal one.*

PROOF Let us suppose that E has no binary decomposition. Its cover corresponds to the tandem $(r, r') = ([i, j], [i', j'])$. Let us consider two different decompositions of E, that correspond respectively to decompositions of the range r in two sets of sub-ranges of the form $[i, k_1], [k_1 + 1, k_2], \ldots, [k_m, j]$ and $[i, k'_1], [k'_1 + 1, k'_2], \ldots, [k'_{m'}, j]$. For simplifying the notations, we write $k_0 = k'_0 = i$ and $k_{m+1} = k_{m'+1} = j$. Since $k_0 = k'_0$, there exist an index $p > 0$ such that for any $l < p$, $k_l = k'_l$, but $k_p \neq k'_p$: p is the index that identifies the first discrepancy between both decomposition. Since $k_{m+1} = k_{m'+1}$, there must exist $q \leq m$ and $q' \leq m'$ such that q and q' are strictly greater than p and that are the minimal indexes such that $k_q = k'_{q'}$. By definition, all bundles of the form $E_{[k_{l-1}, k_l]}$ ($p \leq l \leq q$) have a non-empty intersection with at least one bundle of the form $E_{[k'_{l-1}, k'_l]}$

[3] The term "refinement" is used in the usual way concerning partitions, i.e., a partition P_1 is a refinement of another one P_2 if all constituents in P_1 are constituents of P_2, or belongs to a subset of the partition P_1 that is a partition of one element of P_2.

($p \leq l \leq q'$). The reverse is true as well. Applying Lemma 1, this shows that $E([k_{p+1}, k_q])$ is a bundle with fan-out 2. Therefore, by replacing all ranges involved in this union in one decomposition or the other, we get a third decomposition for which the two initial ones are strict refinements. This is a contradiction, which concludes the proof. ∎

Lemma 2 *Let $E = V(r \cup r')$ be a bundle, with $r = [i, j]$. We suppose it has a unique (non-binary) optimal decomposition, which decomposes $[i, j]$ into $[i, k_1], [k_1 + 1, k_2], \ldots, [k_m, j]$. There exist no range $r'' \subset r$ such that (i) $E_{r''}$ is a bundle and (ii) $\exists l$, $1 \leq l \leq m$ such that $[k_l, k_{l+1}] \subset r''$.*

PROOF Let us consider a range r'' that would contradict the lemma. The union of r'' and of the ranges in the optimal decomposition that have a non-empty intersection with r'' is a fan-out 2 bundle that includes at least two elements of the optimal decomposition, but that is strictly included in E because the decomposition is not binary. This is a contradiction. ∎

Lemma 3 *Let $E = V(r, r')$ be a bundle, with $r = [i, j]$. We suppose it has a binary (optimal) decomposition (not necessarily unique). Let $r'' = [i, k]$ be the largest range starting in i such that $k < j$ and such that it anchors a bundle, namely $E(r'')$. Then $E(r'')$ and $E([k + 1, j])$ form a binary decomposition of E.*

PROOF We need to prove that $E([k + 1, j])$ is a bundle. Each (optimal) binary decomposition of E decomposes r in 1, 2 or 3 sub-ranges. If no optimal decomposition decomposes r in at least 2 sub-ranges, then the proof given here can be adapted by reasoning on r' instead of r. We now suppose that at least one of them decomposes r in at least 2 sub-ranges. Therefore, it decomposes r in $[i, k_1]$ and $[k_1 + 1, j]$ or in $[i, k_1]$, $[k_1 + 1, k_2]$ and $[k_2 + 1, j]$. We select one of these optimal decomposition by taking one such that k_1 is maximal. We shall now distinguish between two cases.

First, let us suppose that r is decomposed into two sub-ranges $[i, k_1]$ and $[k_1 + 1, j]$ by the selected optimal decomposition. Obviously, $E([i, k_1])$ is a "crossing" bundle, i.e., the right component of its cover is is a sub-range of r'. Since r is decomposed in two sub-ranges, it is necessarily the same for r'. Therefore, $E([i, k_1])$ has a cover of the form $[i, k_1] \cup [i', k'_1]$ or $[i, k_1] \cup [k'_1 + 1, j]$. Since r'' includes $[i, k_1]$, $E(r'')$ has a cover of the form $[i, k] \cup [i', k']$ or $[i, k] \cup [k' + 1, j]$. This means that r' is decomposed by $E(r'')$ in only 2 ranges, namely the right component of $E(r'')$'s cover and another range, that we can call r'''. Since $r \setminus r'' = [k + 1, j]$ may not anchor a bundle with fan-out 1, it must contain at least one crossing edge. All such edges necessarily fall within r'''. Conversely, any crossing edge that falls inside r''' necessarily has its other end inside $[k + 1, j]$. Which means that $E(r'')$ and $E(r''')$ form a binary decomposition of E. Therefore, by definition of k_1, $k = k_1$.

Second, let us suppose that r is decomposed into 3 sub-ranges by the selected original decomposition (therefore, r' is not decomposed by this decomposition). This means that this decomposition involves a bundle with a cover of the form $[i, k_1] \cup [k_2 + 1, j]$ and another bundle with a cover of the form $[k_1 + 1, k_2] \cup r'$ (this bundle is in fact $E(r')$). If $k \geq k_2$, then the left range of both members of the original decomposition are included in r'', which means that $E(r'') = E$, and therefore $r'' = r$ which is excluded. Note that k is at least as large as k_1 (since $[i, k_1]$ is a valid "range starting in i such that $k < j$ and such that it anchors a bundle"). Therefore, we have $k_1 \leq k < k_2$. Therefore, $E([i, k_1]) \subset E(r'')$, which means that all edges anchored inside $[k_2 + 1, j])$ are included in $E(r'')$. Hence, $E(r'')$ can not be a crossing bundle without having a left component that is $[i, j]$, which is excluded (it would mean $E(r'') = E$). This means that $E(r'')$ is a bundle with a cover of the form $[i, k] \cup [k' + 1, j]$. Which means that $E(r')$ is in fact the bundle whose cover is $[k + 1, k' + 1] \cup r'$. Hence, $E(r'')$ and $E(r')$ form a binary decomposition of E. Hence, by definition of k_1, $k = k_1$. ∎

As an immediate consequence of Lemmas 2 and 3, our algorithm for extracting the optimal decomposition for F' consists in applying the following procedure recursively, starting with F', and repeating it on each constructed sub-bundle E, until sub-bundles with only one edge are reached.

Let $E = E(r, r')$ be a bundle, with $r = [i, j]$. One optimal decomposition of E can be obtained as follows. One selects the bundle with a left component starting in i and with the maximum length, and iterating this selection process until r is covered. The same is done with r'. We retain the optimal among both resulting decompositions (or one of them if they are both optimal). Note that this

decomposition is unique if and only if it has four components or more; it can not be ternary; it may be binary, and in this case it may be non-unique.

This algorithm gives us a way to extract an optimal decomposition of F' in linear time w.r.t. the number of sub-bundles in this optimal decomposition. The only required data structure is, for each i (resp. k), the list of bundles with a cover of the form $[i,j] \cup [k,l]$ ordered by decreasing j (resp. l). This can trivially be constructed in time $\mathcal{O}(n^2)$ from the list of all bundles we built in time $\mathcal{O}(n^2)$ in the previous section. Since the number of bundles is bounded by $\mathcal{O}(n^2)$ (as mentioned above and proved in Section 5), this means we can extract an optimal decomposition for F' in $\mathcal{O}(n^2)$.

Similar ideas apply to the simpler case of the decomposition of bundles with fan-out 1.

4.6 The main decomposition algorithm

We now have to generalize our algorithm in order to handle the possible existence of fan-out 1 bundles. We achieve this by using the fan-out 2 algorithm recursively. First, we extract and remove (maximal) bundles with fan-out 1 from F, and recursively apply to each of them the complete algorithm. What remains is F', which is a set of bundles with no sub-bundles with fan-out 1. This means we can apply the algorithm presented above. Then, for each bundle with fan-out 1, we group it with a randomly chosen adjacent bundle with fan-out 2, which builds an expanded bundle with fan-out 2, which has a binary decomposition into the original bundle with fan-out 2 and the bundle with fan-out 1.

5 Time complexity analysis

In Section 4, we claimed that there are no more than $\mathcal{O}(n^2)$ bundles. In this section we sketch the proof of this result, which will prove the quadratic time complexity of our algorithm.

Let us compute an upper bound on the number of bundles with fan-out two that can be found within the p-graph processed in Section 4.5, i.e., a p-graph with no fan-out 1 sub-bundle.

Let $E, E' \subseteq E_p$ be bundles with fan-out two. If $E \subset E'$, then we say that E' **expands** E. E' is said to **immediately** expand E, written $E \rightarrow E'$, if E' expands E and there is no bundle E'' such that E'' expands E and E' expands E''.

Let us represent bundles and the associated immediate expansion relation by means of a graph. Let $\mathcal{E}$ denote the set of all bundles (with fan-out two) in our p-graph. The **e-graph** associated with our LCFRS production p is the directed graph with vertices $\mathcal{E}$ and edges defined by the relation $\rightarrow$. For $E \in \mathcal{E}$, we let $out(E) = \{E' \mid E \rightarrow E'\}$ and $in(E) = \{E' \mid E' \rightarrow E\}$.

Lack of space prevents us from providing the proof of the following property. For any $E \in \mathcal{E}$ that contains more than one edge, $|out(E)| \leq 2$ and $|in(E)| \geq 2$. This allows us to prove our upper bound on the size of $\mathcal{E}$.

Theorem 2 *The e-graph associated with an* LCFRS *production p has at most n^2 vertices, where n is the rank of p.*

PROOF Consider the e-graph associated with production p, with set of vertices $\mathcal{E}$. For a vertex $E \in \mathcal{E}$, we define the **level** of E as the number $|E|$ of edges in the corresponding bundle from the p-graph associated with p. Let d be the maximum level of a vertex in $\mathcal{E}$. We thus have $1 \leq d \leq n$. We now prove the following claim. For any integer k with $1 \leq k \leq d$, the set of vertices in $\mathcal{E}$ with level k has no more than n elements.

For $k = 1$, since there are no more than n edges in such a p-graph, the statement holds.

We can now consider all vertices in $\mathcal{E}$ with level $k > 1$ ($k \leq d$). Let $\mathcal{E}^{(k-1)}$ be the set of all vertices in $\mathcal{E}$ with level smaller than or equal to $k-1$, and let us call $T^{(k-1)}$ the set of all edges in the e-graph that are leaving from some vertex in $\mathcal{E}^{(k-1)}$. Since for each bundle E in $\mathcal{E}^{(k-1)}$ we know that $|out(E)| \leq 2$, we have $|T^{(k-1)}| \leq 2|\mathcal{E}^{(k-1)}|$.

The number of vertices in $\mathcal{E}^{(k)}$ with level larger than one is at least $|\mathcal{E}^{(k-1)}| - n$. Since for each $E \in \mathcal{E}^{(k-1)}$ we know that $|in(E)| \geq 2$, we conclude that at least $2(|\mathcal{E}^{(k-1)}| - n)$ edges in $T^{(k-1)}$ must end up at some vertex in $\mathcal{E}^{(k)}$. Let T be the set of edges in $T^{(k-1)}$ that impinge on some vertex in $\mathcal{E} \setminus \mathcal{E}^{(k)}$. Thus we have $|T| \leq 2|\mathcal{E}^{(k-1)}| - 2(|\mathcal{E}^{(k-1)}| - n) = 2n$. Since the vertices of level k in $\mathcal{E}$ must have incoming edges from set T, and because each of them have at least 2 incoming edges, there cannot be more than n such vertices. This concludes the proof of our claim.

Since the the level of a vertex in $\mathcal{E}$ is necessarily lower than n, this completes the proof. ∎

The overall complexity of the complete algorithm can be computed by induction. Our induction hypothesis is that for $m < n$, the time complexity is in $\mathcal{O}(m^2)$. This is obviously true for $n = 1$ and $n = 2$. Extracting the bundles

with fan-out 1 costs $\mathcal{O}(n^2)$. These bundles are of length $n_1 \dots n_m$. Extracting bundles with fan-out 2 costs $\mathcal{O}((n - n_1 - \dots - n_m)^2)$. Applying recursively the algorithm to bundles with fan-out 1 costs $\mathcal{O}(n_1^2) + \dots + \mathcal{O}(n_m^2)$. Therefore, the complexity is in $\mathcal{O}(n^2) + \mathcal{O}((n - n_1 - \dots - n_m)^2) + \sum_{i=1}^{n} \mathcal{O}(n_i) = \mathcal{O}(n^2) + \mathcal{O}(\sum_{i=1}^{n} n_i) = \mathcal{O}(n^2)$.

6 Conclusion

We have introduced an efficient algorithm for optimal reduction of the rank of LCFRSs with fan-out at most 2, that runs in quadratic time w.r.t. the rank of the input grammar. Given the fact that fan-out 1 bundles can be attached to any adjacent bundle in our factorization, we can show that our algorithm also optimizes time complexity for known tabular parsing algorithms for LCFRSs with fan-out 2.

As for general LCFRS, it has been shown by Gildea (2010) that rank optimization and time complexity optimization are not equivalent. Furthermore, all known algorithms for rank or time complexity optimization have an exponential time complexity (Gómez-Rodríguez et al., 2009).

Acknowledgments

Part of this work was done while the second author was a visiting scientist at Alpage (INRIA Paris-Rocquencourt and Université Paris 7), and was financially supported by the hosting institutions.

References

Daniel Gildea. 2010. Optimal parsing strategies for linear context-free rewriting systems. In *Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, Los Angeles, California. To appear.

Carlos Gómez-Rodríguez and Giorgio Satta. 2009. An optimal-time binarization algorithm for linear context-free rewriting systems with fan-out two. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 985–993, Suntec, Singapore, August. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta, and David J. Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL'09:HLT)*, Boulder, Colorado. To appear.

Michael A. Harrison. 1978. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, MA.

Aravind K. Joshi and Leon S. Levy. 1977. Constraints on local descriptions: Local transformations. *SIAM Journal of Computing*

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, Athens, Greece. To appear.

Wolfgang Maier and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks. In *Proceedings of the 14th Conference on Formal Grammar (FG 2009)*, Bordeaux, France.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In Philippe de Groote, editor, *Proceedings of the 13th Conference on Formal Grammar (FG 2008)*, pages 61–76, Hamburg, Germany. CSLI Publications.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:87–120.

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics (ACL'87)*.

David J. Weir. 1992. Linear context-free rewriting systems and deterministic tree-walk transducers. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.

The Importance of Rule Restrictions in CCG

Marco Kuhlmann
Dept. of Linguistics and Philology
Uppsala University
Uppsala, Sweden

Alexander Koller
Cluster of Excellence
Saarland University
Saarbrücken, Germany

Giorgio Satta
Dept. of Information Engineering
University of Padua
Padua, Italy

Abstract

Combinatory Categorial Grammar (CCG) is generally construed as a fully lexicalized formalism, where all grammars use one and the same universal set of rules, and cross-linguistic variation is isolated in the lexicon. In this paper, we show that the weak generative capacity of this 'pure' form of CCG is strictly smaller than that of CCG with grammar-specific rules, and of other mildly context-sensitive grammar formalisms, including Tree Adjoining Grammar (TAG). Our result also carries over to a multi-modal extension of CCG.

1 Introduction

Combinatory Categorial Grammar (CCG) (Steedman, 2001; Steedman and Baldridge, 2010) is an expressive grammar formalism with formal roots in combinatory logic (Curry et al., 1958) and links to the type-logical tradition of categorial grammar (Moortgat, 1997). It has been successfully used for a wide range of practical tasks, such as data-driven parsing (Hockenmaier and Steedman, 2002; Clark and Curran, 2007), wide-coverage semantic construction (Bos et al., 2004), and the modelling of syntactic priming (Reitter et al., 2006).

It is well-known that CCG can generate languages that are not context-free (which is necessary to capture natural languages), but can still be parsed in polynomial time. Specifically, Vijay-Shanker and Weir (1994) identified a version of CCG that is weakly equivalent to Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) and other mildly context-sensitive grammar formalisms, and can generate non-context-free languages such as $a^n b^n c^n$. The generative capacity of CCG is commonly attributed to its flexible *composition* rules, which allow it to model more complex word orders that context-free grammar can.

The discussion of the (weak and strong) generative capacity of CCG and TAG has recently been revived (Hockenmaier and Young, 2008; Koller and Kuhlmann, 2009). In particular, Koller and Kuhlmann (2009) have shown that CCGs that are *pure* (i.e., they can only use generalized composition rules, and there is no way to restrict the instances of these rules that may be used) and *first-order* (i.e., all argument categories are atomic) can *not* generate $a^n b^n c^n$. This shows that the generative capacity of at least first-order CCG crucially relies on its ability to restrict rule instantiations, and is at odds with the general conception of CCG as a fully lexicalized formalism, in which all grammars use one and the same set of universal rules. A question then is whether the result carries over to pure CCG with higher-order categories.

In this paper, we answer this question to the positive: We show that the weak generative capacity of general pure CCG is still strictly smaller than that of the formalism considered by Vijay-Shanker and Weir (1994); composition rules can only achieve their full expressive potential if their use can be restricted. Our technical result is that every language L that can be generated by a pure CCG has a context-free sublanguage $L' \subseteq L$ such that every string in L is a permutation of a string in L', and vice versa. This means that $a^n b^n c^n$, for instance, cannot be generated by pure CCG, as it does not have any (non-trivial) permutation-equivalent sublanguages. Conversely, we show that there are still languages that can be generated by pure CCG but not by context-free grammar.

We then show that our permutation language lemma also holds for pure *multi-modal* CCG as defined by Baldridge and Kruijff (2003), in which the use of rules can be controlled through the lexicon entries by assigning types to slashes. Since this extension was intended to do away with the need for grammar-specific rule restrictions, it comes as quite a surprise that pure multi-modal

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 534–543,
Uppsala, Sweden, 11-16 July 2010.

CCG in the style of Baldridge and Kruijff (2003) is still less expressive than the CCG formalism used by Vijay-Shanker and Weir (1994). This means that word order in CCG cannot be fully lexicalized with the current formal tools; some ordering constraints must be specified via language-specific combination rules and not in lexicon entries. On the other hand, as pure multi-modal CCG has been successfully applied to model the syntax of a variety of natural languages, another way to read our results is as contributions to a discussion about the exact expressiveness needed to model natural language.

The remainder of this paper is structured as follows. In Section 2, we introduce the formalism of *pure* CCG that we consider in this paper, and illustrate the relevance of rule restrictions. We then study the generative capacity of pure CCG in Section 3; this section also presents our main result. In Section 4, we show that this result still holds for multi-modal CCG. Section 5 concludes the paper with a discussion of the relevance of our findings.

2 Combinatory Categorial Grammar

We start by providing formal definitions for categories, syntactic rules, and grammars, and then discuss the relevance of rule restrictions for CCG.

2.1 Categories

Given a finite set A of *atomic categories*, the *set of categories over* A is the smallest set C such that $A \subseteq C$, and $(\mathsf{x}/\mathsf{y}), (\mathsf{x}\backslash\mathsf{y}) \in C$ whenever $\mathsf{x}, \mathsf{y} \in C$. A category x/y represents a function that seeks a string with category y to the right (indicated by the forward slash) and returns a new string with category x; a category $\mathsf{x}\backslash\mathsf{y}$ instead seeks its argument to the left (indicated by the backward slash). In the remainder of this paper, we use lowercase sans-serif letters such as $\mathsf{x}, \mathsf{y}, \mathsf{z}$ as variables for categories, and the vertical bar $|$ as a variable for slashes. In order to save some parentheses, we understand slashes as left-associative operators, and write a category such as $(\mathsf{x}/\mathsf{y})\backslash\mathsf{z}$ as $\mathsf{x}/\mathsf{y}\backslash\mathsf{z}$.

The list of *arguments* of a category c is defined recursively as follows: If c is atomic, then it has no arguments. If $\mathsf{c} = \mathsf{x}|\mathsf{y}$ for some categories x and y, then the arguments of c are the slashed category $|\mathsf{y}$, plus the arguments of x. We number the arguments of a category from outermost to innermost. The *arity* of a category is the number of its arguments. The *target* of a category c is the atomic category that remains when stripping c of its arguments.

x/y	y	$\Rightarrow$	x	forward application $>$
y	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	x	backward application $<$
x/y	y/z	$\Rightarrow$	x/z	forward harmonic composition $>\mathsf{B}$
$\mathsf{y}\backslash\mathsf{z}$	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	$\mathsf{x}\backslash\mathsf{z}$	backward harmonic composition $<\mathsf{B}$
x/y	$\mathsf{y}\backslash\mathsf{z}$	$\Rightarrow$	$\mathsf{x}\backslash\mathsf{z}$	forward crossed composition $>\mathsf{B}_\times$
y/z	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	x/z	backward crossed composition $<\mathsf{B}_\times$

Figure 1: The core set of rules of CCG.

2.2 Rules

The syntactic rules of CCG are directed versions of combinators in the sense of combinatory logic (Curry et al., 1958). Figure 1 lists a core set of commonly assumed rules, derived from functional application and the B combinator, which models functional composition. When talking about these rules, we refer to the premise containing the argument $|\mathsf{y}$ as the *primary premise*, and to the other premise as the *secondary premise* of the rule.

The rules in Figure 1 can be generalized into composition rules of higher degrees. These are defined as follows, where $n \geq 0$ and β is a variable for a sequence of n arguments.

x/y	$\mathsf{y}\beta$	$\Rightarrow$	$\mathsf{x}\beta$	generalized forward composition $>^n$
$\mathsf{y}\beta$	$\mathsf{x}\backslash\mathsf{y}$	$\Rightarrow$	$\mathsf{x}\beta$	generalized backward composition $<^n$

We call the value n the *degree* of the composition rule. Note that the rules in Figure 1 are the special cases for $n = 0$ and $n = 1$.

Apart from the core rules given in Figure 1, some versions of CCG also use rules derived from the S and T combinators of combinatory logic, called *substitution* and *type-raising*, the latter restricted to the lexicon. However, since our main point of reference in this paper, the CCG formalism defined by Vijay-Shanker and Weir (1994), does *not* use such rules, we will not consider them here, either.

2.3 Grammars and Derivations

With the set of rules in place, we can define a *pure combinatory categorial grammar* (PCCG) as a construct $G = (A, \Sigma, L, \mathsf{s})$, where A is an alphabet of *atomic categories*, $\mathsf{s} \in A$ is a distinguished atomic category called the *final category*, Σ is a finite set of *terminal symbols*, and L is a finite relation between symbols in Σ and categories over A, called the *lexicon*. The elements of the lexicon L are called *lexicon entries*, and we represent them using the notation $\sigma \vdash \mathsf{x}$, where $\sigma \in \Sigma$ and x is a category over A. A category that occurs in a lexicon entry is called a *lexical category*.

A derivation in a grammar G can be represented as a *derivation tree* as follows. Given a string $w \in \Sigma^*$, we choose a lexicon entry for each occurrence of a symbol in w, line up the respective lexical categories from left to right, and apply admissible rules to adjacent pairs of categories. After the application of a rule, only the conclusion is available for future applications. We iterate this process until we end up with a single category. The string w is called the *yield* of the resulting derivation tree. A derivation tree is *complete*, if the last category is the final category of G. The *language generated by* G, denoted by $L(G)$, is formed by the yields of all complete derivation trees.

2.4 Degree Restrictions

Work on CCG generally assumes an upper bound on the degree of composition rules that can be used in derivations. We also employ this restriction, and only consider grammars with compositions of some bounded (but arbitrary) degree $n \geq 0$.[1] CCG with unbounded-degree compositions is more expressive than bounded-degree CCG or TAG (Weir and Joshi, 1988).

Bounded-degree grammars have a number of useful properties, one of which we mention here. The following lemma rephrases Lemma 3.1 in Vijay-Shanker and Weir (1994).

Lemma 1 *For every grammar G, every argument in a derivation of G is the argument of some lexical category of G.*

As a consequence, there is only a finite number of categories that can occur as arguments in some derivation. In the presence of a bound on the degree of composition rules, this implies the following:

Lemma 2 *For every grammar G, there is a finite number of categories that can occur as secondary premises in derivations of G.*

Proof. The arity of a secondary premise c can be written as $m + n$, where m is the arity of the first argument of the corresponding primary premise, and n is the degree of the rule applied. Since each argument is an argument of some lexical category of G (Lemma 1), and since n is assumed to be bounded, both m and n are bounded. Hence, there is a bound on the number of choices for c. ∎

Note that the number of categories that can occur as *primary* premises is generally unbounded even in a grammar with bounded degree.

[1] For practical grammars, $n \leq 4$.

2.5 Rule Restrictions

The rule set of pure CCG is universal: the difference between the grammars of different languages should be restricted to different choices of categories in the lexicon. This is what makes pure CCG a lexicalized grammar formalism (Steedman and Baldridge, 2010). However, most practical CCG grammars rely on the possibility to exclude or restrict certain rules. For example, Steedman (2001) bans the rule of forward crossed composition from his grammar of English, and stipulates that the rule of backward crossed composition may be applied only if both of its premises share the common target category s, representing sentences. Exclusions and restrictions of rules are also assumed in much of the language-theoretic work on CCG. In particular, they are essential for the formalism used in the aforementioned equivalence proof for CCG and TAG (Vijay-Shanker and Weir, 1994).

To illustrate the formal relevance of rule restrictions, suppose that we wanted to write a pure CCG that generates the language

$$L_3 \;=\; \{\, a^n b^n c^n \mid n \geq 1 \,\},$$

which is not context-free. An attempt could be

$$G_1 \;=\; (\{\mathsf{s},\mathsf{a},\mathsf{b},\mathsf{c}\},\{a,b,c\},L,\mathsf{s}),$$

where the lexicon L is given as follows:

$$a \vdash \mathsf{a}\,,\; b \vdash \mathsf{s/c\backslash a}\,,\; b \vdash \mathsf{b/c\backslash a}\,,$$
$$b \vdash \mathsf{s/c/b\backslash a}\,,\; b \vdash \mathsf{s/c/b\backslash a}\,,\; c \vdash \mathsf{c}\,.$$

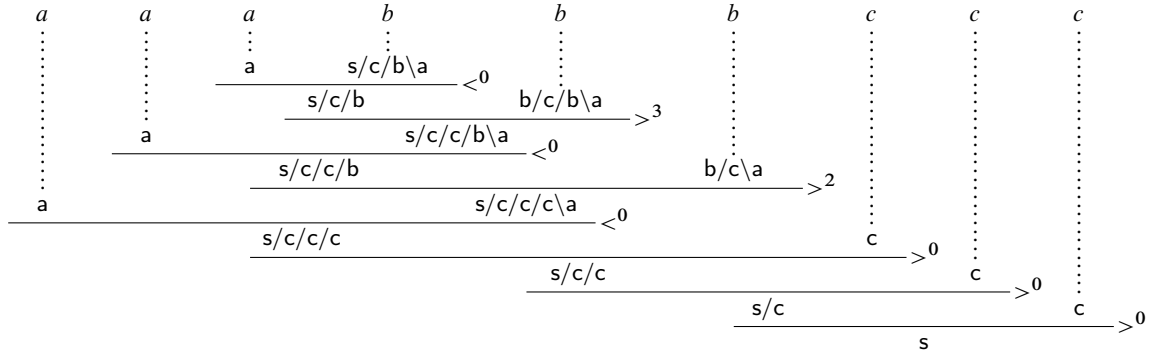
From a few sample derivations like the one given in Figure 2a, we can convince ourselves that G_1 generates all strings of the form $a^n b^n c^n$, for any $n \geq 1$. However, a closer inspection reveals that it also generates other, unwanted strings—in particular, strings of the form $(ab)^n c^n$, as witnessed by the derivation given in Figure 2b.

Now suppose that we would have a way to only allow those instances of generalized composition in which the secondary premise has the form $\mathsf{b/c/b\backslash a}$ or $\mathsf{b/c\backslash a}$. Then the compositions

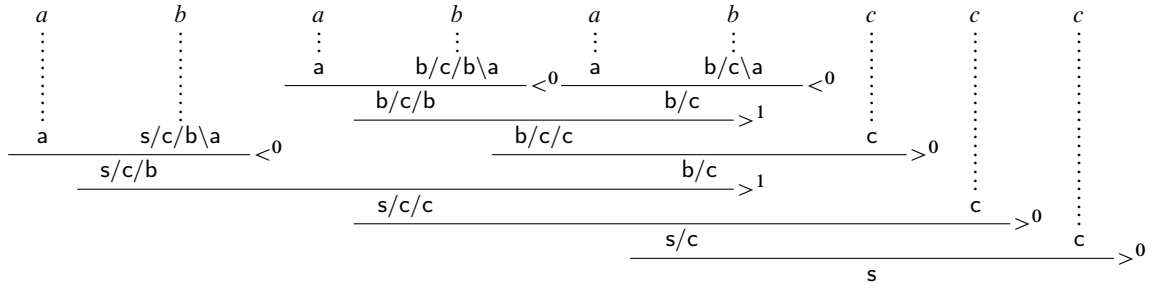
$$\frac{\mathsf{b/c/b}\quad\mathsf{b/c}}{\mathsf{b/c/c}}\,{>}^1 \quad\text{and}\quad \frac{\mathsf{s/c/b}\quad\mathsf{b/c}}{\mathsf{s/c/c}}\,{>}^1$$

would be disallowed, and it is not hard to see that G_1 would generate exactly $a^n b^n c^n$.

As we will show in this paper, our attempt to capture L_3 with a pure CCG grammar failed not only because we could not think of one: L_3 cannot be generated by *any* pure CCG.



(a) Derivation of the string $aaabbbccc$.



(b) Derivation of the string $abababccc$.

Figure 2: Two derivations of the grammar G_1.

3 The Generative Capacity of Pure CCG

We will now develop a formal argument showing that rule restrictions increase the weak generative capacity of CCG. We will first prove that pure CCG is still more expressive than context-free grammar. We will then spend the rest of this section working towards the result that pure CCG is strictly less expressive than CCG with rule restrictions. Our main technical result will be the following:

Theorem 1 *Every language that can be generated by a pure CCG has a Parikh-equivalent context-free sublanguage.*

Here, two languages L and L' are called *Parikh-equivalent* if every string in L is the permutation of a string in L' and vice versa.

3.1 CFG $\subsetneq$ PCCG

Proposition 1 *The class of languages generated by pure CCG properly includes the class of context-free languages.*

Proof. To see the inclusion, it suffices to note that pure CCG when restricted to application rules is the same as *AB-grammar*, the classical categorial formalism investigated by Ajdukiewicz and Bar-Hillel (Bar-Hillel et al., 1964). This formalism is weakly equivalent to context-free grammar.

To see that the inclusion is proper, we can go back to the grammar G_1 that we gave in Section 2.5. We have already discussed that the language L_3 is included in $L(G_1)$. We can also convince ourselves that all strings generated by the grammar G_1 have an equal number of as, bs and cs. Consider now the regular language $R = a^*b^*c^*$. From our observations, it follows that $L(G_1) \cap R = L_3$. Since context-free languages are closed under intersection with regular languages, we find that $L(G_1)$ can be context-free only if L_3 is. Since L_3 is not context-free, we therefore conclude that $L(G_1)$ is not context-free, either. ∎

Two things are worth noting. First, our result shows that the ability of CCG to generate non-context-free languages does not hinge on the availability of substitution and type-raising rules: The derivations of G_1 only use generalized compositions. Neither does it require the use of functional argument categories: The grammar G_1 is *first-order* in the sense of Koller and Kuhlmann (2009).

Second, it is important to note that if the composition degree n is restricted to 0 or 1, pure CCG actually collapses to context-free expressive power. This is clear for $n = 0$ because of the equivalence to AB grammar. For $n = 1$, observe that the arity of the result of a composition is at most as high as

that of each premise. This means that the arity of any derived category is bounded by the maximal arity of lexical categories in the grammar, which together with Lemma 1 implies that there is only a finite set of derivable categories. The set of all valid derivations can then be simulated by a context-free grammar. In the presence of rules with $n \geq 2$, the arities of derived categories can grow unboundedly.

3.2 Active and Inactive Arguments

In the remainder of this section, we will develop the proof of Theorem 1, and use it to show that the generative capacity of PCCG is strictly smaller than that of CCG with rule restrictions. For the proof, we adopt a certain way to view the information flow in CCG derivations. Consider the following instance of forward harmonic composition:

$$\mathsf{a/b}\ \ \mathsf{b/c}\ \Rightarrow\ \mathsf{a/c}$$

This rule should be understood as obtaining its conclusion $\mathsf{a/c}$ from the primary premise $\mathsf{a/b}$ by the removal of the argument $\mathsf{/b}$ and the subsequent transfer of the argument $\mathsf{/c}$ from the secondary premise. With this picture in mind, we will view the two occurrences of $\mathsf{/c}$ in the secondary premise and in the conclusion as two occurrences of one and the same argument. Under this perspective, in a given derivation, an argument has a lifespan that starts in a lexical category and ends in one of two ways: either in the primary or in the secondary premise of a composition rule. If it ends in a primary premise, it is because it is matched against a subcategory of the corresponding secondary premise; this is the case for the argument $\mathsf{/b}$ in the example above. We will refer to such arguments as *active*. If an argument ends its life in a secondary premise, it is because it is consumed as part of a higher-order argument. This is the case for the argument $\mathsf{/c}$ in the secondary premise of the following rule instance:

$$\mathsf{a/(b/c)}\ \ \mathsf{b/c/d}\ \Rightarrow\ \mathsf{a/d}$$

(Recall that we assume that slashes are left-associative.) We will refer to such arguments as *inactive*. Note that the status of an argument as either active or inactive is not determined by the grammar, but depends on a concrete derivation.

The following lemma states an elementary property in connection with active and inactive arguments, which we will refer to as *segmentation*:

Lemma 3 *Every category that occurs in a CCG derivation has the general form* $\mathsf{a}\alpha\beta$*, where* a *is an atomic category,* α *is a sequence of inactive arguments, and* β *is a sequence of active arguments.*

Proof. The proof is by induction on the depth of a node in the derivation. The property holds for the root (which is labeled with the final category), and is transferred from conclusions to premises. ∎

3.3 Transformation

The fundamental reason for why the example grammar G_1 from Section 2.5 overgenerates is that in the absence of rule restrictions, we have no means to control the point in a derivation at which a category combines with its arguments. Consider the examples in Figure 2: It is because we cannot ensure that the bs finish combining with the other bs before combining with the cs that the undesirable word order in Figure 2b has a derivation. To put it as a slogan: Permuting the words allows us to saturate arguments prematurely.

In this section, we show that this property applies to all pure CCGs. More specifically, we show that, in a derivation of a pure CCG, almost all active arguments of a category can be saturated before that category is used as a secondary premise; at most one active argument must be transferred to the conclusion of that premise. Conversely, any derivation that still contains a category with at least two active arguments can be transformed into a new derivation that brings us closer to the special property just characterized.

We formalize this transformation by means of a system of rewriting rules in the sense of Baader and Nipkow (1998). The rules are given in Figure 3. To see how they work, let us consider the first rule, R1; the other ones are symmetric. This rules states that, whenever we see a derivation in which a category of the form $\mathsf{x/y}$ (here marked as **A**) is combined with a category of the form $\mathsf{y}\beta\mathsf{/z}$ (marked as **B**), and the result of this combination is combined with a category of the form $\mathsf{z}\gamma$ (**C**), then the resulting category can also be obtained by 'rotating' the derivation to first saturate $\mathsf{/z}$ by combining **B** with **C**, and only then do the combination with **A**. When applying these rotations exhaustively, we end up with a derivation in which almost all active arguments of a category are saturated before that category is used as a secondary premise. Applying the transformation to the derivation in Figure 2a, for instance, yields the derivation in Figure 2b.

We need the following result for some of the lemmas we prove below. We call a node in a deriv-

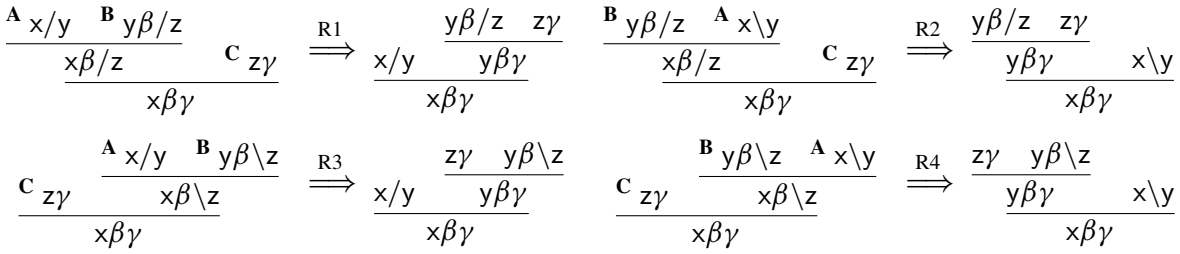


Figure 3: Rewriting rules used in the transformation. Here, γ represents a (possibly empty) sequence of arguments, and β represents a sequence of arguments in which the first (outermost) argument is active.

ation *critical* if its corresponding category contains more than one active argument and it is the secondary premise of a rule. We say that u is a *highest* critical node if there is no other critical node whose distance to the root is shorter.

Lemma 4 *If u is a highest critical node, then we can apply one of the transformation rules to the grandparent of u.*

Proof. Suppose that the category at u has the form $\mathsf{y}\beta/\mathsf{z}$, where $/\mathsf{z}$ is an active argument, and the first argument in β is active as well. (The other possible case, in which the relevant occurrence has the form $\mathsf{y}\beta\backslash\mathsf{z}$, can be treated symmetrically.) Since u is a secondary premise, it is involved in an inference of one of the following two forms:

$$\frac{\mathsf{x}/\mathsf{y} \quad \mathsf{y}\beta/\mathsf{z}}{\mathsf{x}\beta/\mathsf{z}} \qquad \frac{\mathsf{y}\beta/\mathsf{z} \quad \mathsf{x}\backslash\mathsf{y}}{\mathsf{x}\beta/\mathsf{z}}$$

Since u is a highest critical node, the conclusion of this inference is not a critical node itself; in particular, it is not a secondary premise. Therefore, the above inferences can be extended as follows:

$$\frac{\dfrac{\mathsf{x}/\mathsf{y} \quad \mathsf{y}\beta/\mathsf{z}}{\mathsf{x}\beta/\mathsf{z}} \quad \mathsf{z}\gamma}{\mathsf{x}\beta\gamma} \qquad \frac{\dfrac{\mathsf{y}\beta/\mathsf{z} \quad \mathsf{x}\backslash\mathsf{y}}{\mathsf{x}\beta/\mathsf{z}} \quad \mathsf{z}\gamma}{\mathsf{x}\beta\gamma}$$

These partial derivations match the left-hand side of the rewriting rules R1 and R2, respectively. Hence, we can apply a rewriting rule to the derivation. ∎

We now show that the transformation is well-defined, in the sense that it terminates and transforms derivations of a grammar G into new derivations of G.

Lemma 5 *The rewriting of a derivation tree ends after a finite number of steps.*

Proof. We assign natural numbers to the nodes of a derivation tree as follows. Each leaf node is assigned the number 0. For an inner node u, which corresponds to the conclusion of a composition rule, let m, n be the numbers assigned to the nodes corresponding to the primary and secondary premise, respectively. Then u is assigned the number $1 + 2m + n$. Suppose now that we have associated premise **A** with the number x, premise **B** with the number y, and premise **C** with the number z. It is then easy to verify that the conclusion of the partial derivation on the left-hand side of each rule has the value $3 + 4x + 2y + z$, while the conclusion of the right-hand side has the value $2 + 2x + 2y + z$. Thus, each step decreases the value of a derivation tree under our assignment by the amount $1 + 2x$. Since this value is positive for all choices of x, the rewriting ends after a finite number of steps. ∎

To convince ourselves that our transformation does not create ill-formed derivations, we need to show that none of the rewriting rules necessitates the use of composition operations whose degree is higher than the degree of the operations used in the original derivation.

Lemma 6 *Applying the rewriting rules from the top down does not increase the degree of the composition operations.*

Proof. The first composition rule used in the left-hand side of each rewriting rule has degree $|\beta| + 1$, the second rule has degree $|\gamma|$; the first rule used in the right-hand side has degree $|\gamma|$, the second rule has degree $|\beta| + |\gamma|$. To prove the claim, it suffices to show that $|\gamma| \leq 1$. This is a consequence of the following two observations.

1. In the category $\mathsf{x}\beta\gamma$, the arguments in γ occur on top of the arguments in β, the first of which is active. Using the segmentation property stated in Lemma 3, we can therefore infer that γ does not contain any inactive arguments.

2. Because we apply rules top-down, premise **B** is a highest critical node in the derivation (by Lemma 4). This means that the category at premise **C** contains at most one active argument; otherwise, premise **C** would be a critical node closer to the root than premise **B**. ∎

We conclude that, if we rewrite a derivation d of G top-down until exhaustion, then we obtain a new valid derivation d'. We call all derivations d' that we can build in this way *transformed*. It is easy to see that a derivation is transformed if and only if it contains no critical nodes.

3.4 Properties of Transformed Derivations

The special property established by our transformation has consequences for the generative capacity of pure CCG. In particular, we will now show that the set of all transformed derivations of a given grammar yields a context-free language. The crucial lemma is the following:

Lemma 7 *For every grammar G, there is some $k \geq 0$ such that no category in a transformed derivation of G has arity greater than k.*

Proof. The number of inactive arguments in the primary premise of a rule does not exceed the number of inactive arguments in the conclusion. In a transformed derivation, a symmetric property holds for active arguments: Since each secondary premise contains at most one active argument, the number of active arguments in the conclusion of a rule is not greater than the number of active arguments in its primary premise. Taken together, this implies that the arity of a category that occurs in a transformed derivation is bounded by the sum of the maximal arity of a lexical category (which bounds the number of active arguments), and the maximal arity of a secondary premise (which bounds the number of inactive arguments). Both of these values are bounded in G. ∎

Lemma 8 *The yields corresponding to the set of all transformed derivations of a pure CCG form a context-free language.*

Proof. Let G be a pure CCG. We construct a context-free grammar G_T that generates the yields of the set of all transformed derivations of G.

As the set of terminals of G_T, we use the set of terminals of G. To form the set of nonterminals, we take all categories that can occur in a transformed derivation of G, and mark each argument as either 'active' ($+$) or 'inactive' ($-$), in all possible ways that respect the segmentation property stated in Lemma 3. Note that, because of Lemma 7 and Lemma 1, the set of nonterminals is finite. As the start symbol, we use s, the final category of G.

The set of productions of G_T is constructed as follows. For each lexicon entry $\sigma \vdash \mathsf{c}$ of G, we include all productions of the form $x \to \sigma$, where x is some marked version of c. These productions represent all valid guesses about the activity of the arguments of c during a derivation of G. The remaining productions encode all valid instantiations of composition rules, keeping track of active and inactive arguments to prevent derivations with critical nodes. More specifically, they have the form

$$\mathsf{x}\beta \to \mathsf{x}/\mathsf{y}^+\ \mathsf{y}\beta \quad \text{or} \quad \mathsf{x}\beta \to \mathsf{y}\beta\ \mathsf{x}\backslash\mathsf{y}^+\,,$$

where the arguments in the y-part of the secondary premise are all marked as inactive, the sequence β contains at most one argument marked as active, and the annotations of the left-hand side nonterminal are copied over from the corresponding annotations on the right-hand side.

The correctness of the construction of G_T can be proved by induction on the length of a transformed derivation of G on the one hand, and the length of a derivation of G_T on the other hand. ∎

3.5 PCCG $\subsetneq$ CCG

We are now ready to prove our main result, repeated here for convenience.

Theorem 1 *Every language that can be generated by a pure CCG grammar has a Parikh-equivalent context-free sublanguage.*

Proof. Let G be a pure CCG, and let L_T be the set of yields of the transformed derivations of G. Inspecting the rewriting rules, it is clear that every string of $L(G)$ is the permutation of a string in L_T: the transformation only rearranges the yields. By Lemma 8, we also know that L_T is context-free. Since every transformed derivation is a valid derivation of G, we have $L_T \subseteq L(G)$. ∎

As an immediate consequence, we find:

Proposition 2 *The class of languages generated by pure CCG cannot generate all languages that can be generated by CCG with rule restrictions.*

Proof. The CCG formalism considered by Vijay-Shanker and Weir (1994) can generate the non-context-free language L_3. However, the only Parikh-equivalent sublanguage of that language is L_3 itself. From Theorem 1, we therefore conclude that L_3 cannot be generated by pure CCG. ∎

In the light of the equivalence result established by Vijay-Shanker and Weir (1994), this means that pure CCG cannot generate all languages that can be generated by TAG.

4 Multi-Modal CCG

We now extend Theorem 1 to multi-modal CCG. We will see that at least for a popular version of multi-modal CCG, the B&K-CCG formalism presented by Baldridge and Kruijff (2003), the proof can be adapted quite straightforwardly. This means that even B&K-CCG becomes less expressive when rule restrictions are disallowed.

4.1 Multi-Modal CCG

The term 'multi-modal CCG' (MM-CCG) refers to a family of extensions to CCG which attempt to bring some of the expressive power of Categorial Type Logic (Moortgat, 1997) into CCG. Slashes in MM-CCG have *slash types*, and rules can be restricted to only apply to arguments that have slashes of the correct type. The idea behind this extension is that many constraints that in ordinary CCG can only be expressed in terms of rule restrictions can now be specified in the lexicon entries by giving the slashes the appropriate types.

The most widely-known version of multi-modal CCG is the formalism defined by Baldridge and Kruijff (2003) and used by Steedman and Baldridge (2010); we refer to it as B&K-CCG. This formalism uses an inventory of four slash types, $\{\star, \times, \diamond, \cdot\}$, arranged in a simple type hierarchy: $\star$ is the most general type, $\cdot$ the most specific, and $\times$ and $\diamond$ are in between. Every slash in a B&K-CCG lexicon is annotated with one of these slash types.

The combinatory rules in B&K-CCG, given in Figure 4, are defined to be sensitive to the slash types. In particular, slashes with the types $\diamond$ and $\times$ can only be eliminated by harmonic and crossed compositions, respectively.[2] Thus, a grammar writer can constrain the application of harmonic and crossed composition rules to certain categories by assigning appropriate types to the slashes of this category in the lexicon. Application rules apply to slashes of any type. As before, we call an MM-CCG grammar *pure* if it only uses application and generalized compositions, and does not provide means to restrict rule applications.

[2] Our definitions of generalized harmonic and crossed composition are the same as the ones used by Hockenmaier and Young (2008), but see the discussion in Section 4.3.

$x/_\star y \;\; y \Rightarrow x$		forward application
$y \;\; x\backslash_\star y \Rightarrow x$		backward application
$x/_\diamond y \;\; y/_\diamond z\beta \Rightarrow x/_\diamond z\beta$		forward harmonic composition
$x/_\times y \;\; y\backslash_\times z\beta \Rightarrow x\backslash_\times z\beta$		forward crossed composition
$y\backslash_\diamond z\beta \;\; x\backslash_\diamond y \Rightarrow x\backslash_\diamond z\beta$		backward harmonic composition
$y/_\times z\beta \;\; x\backslash_\times y \Rightarrow x/_\times z\beta$		backward crossed composition

Figure 4: Rules in B&K-CCG.

4.2 Rule Restrictions in B&K-CCG

We will now see what happens to the proof of Theorem 1 in the context of pure B&K-CCG. There is only one point in the entire proof that could be damaged by the introduction of slash types, and that is the result that if a transformation rule from Figure 3 is applied to a correct derivation, then the result is also grammatical. For this, it must not only be the case that the degree on the composition operations is preserved (Lemma 6), but also that the transformed derivation remains consistent with the slash types. Slash types make the derivation process sensitive to word order by restricting the use of compositions to categories with the appropriate type, and the transformation rules permute the order of the words in the string. There is a chance therefore that a transformed derivation might not be grammatical in B&K-CCG.

We now show that this does not actually happen, for rule R3; the other three rules are analogous. Using s_1, s_2, s_3 as variables for the relevant slash types, rule R3 appears in B&K-CCG as follows:

$$\dfrac{z\gamma \quad \dfrac{x/_{s_1} y \quad y|_{s_2} w\beta\backslash_{s_3} z}{x|_{s_2} w\beta\backslash_{s_3} z}}{x|_{s_2} w\beta\gamma} \overset{\text{R3}}{\Longrightarrow} \dfrac{x/_{s_1} y \quad \dfrac{z\gamma \quad y|_{s_2} w\beta\backslash_{s_3} z}{y|_{s_2} w\beta\gamma}}{x|_{s_2} w\beta\gamma}$$

Because the original derivation is correct, we know that, if the slash of w is forward, then s_1 and s_2 are subtypes of $\diamond$; if the slash is backward, they are subtypes of $\times$. A similar condition holds for s_3 and the first slash in γ; if γ is empty, then s_3 can be anything because the second rule is an application.

After the transformation, the argument $/_{s_1} y$ is used to compose with $y|_{s_2} w\beta\gamma$. The direction of the slash in front of the w is the same as before, so the (harmonic or crossed) composition is still compatible with the slash types s_1 and s_2. An analogous argument shows that the correctness of combining $\backslash_{s_3} z$ with γ carries over from the left to the right-hand side. Thus the transformation maps grammatical derivations into grammatical derivations. The rest of the proof in Section 3 continues to work literally, so we have the following result:

Theorem 2 *Every language that can be generated by a pure B&K-CCG grammar contains a Parikh-equivalent context-free sublanguage.*

This means that pure B&K-CCG is just as unable to generate L_3 as pure CCG is. In other words, the weak generative capacity of CCG with rule restrictions, and in particular that of the formalism considered by Vijay-Shanker and Weir (1994), is strictly greater than the generative capacity of pure B&K-CCG—although we conjecture (but cannot prove) that pure B&K-CCG is still more expressive than pure non-modal CCG.

4.3 Towards More Expressive MM-CCGs

To put the result of Theorem 2 into perspective, we will now briefly consider ways in which B&K-CCG might be modified in order to obtain a pure multi-modal CCG that is weakly equivalent to CCG in the style of Vijay-Shanker and Weir (1994). Such a modification would have to break the proof in Section 4.2, which is harder than it may seem at first glance. For instance, simply assuming a more complex type system will not do it, because the arguments $\backslash_{s_3}\mathsf{z}$ and $/_{s_1}\mathsf{y}$ are eliminated using the same rules in the original and the transformed derivations, so if the derivation step was valid before, it will still be valid after the transformation. Instead, we believe that it is necessary to make the composition rules sensitive to the categories inside β and γ instead of only the arguments $\backslash_{s_3}\mathsf{z}$ and $/_{s_1}\mathsf{y}$, and we can see two ways how to do this.

First, one could imagine a version of multi-modal CCG with unary modalities that can be used to mark certain category occurrences. In such an MM-CCG, the composition rules for a certain slash type could be made sensitive to the presence or absence of unary modalities in β. Say for instance that the slash type s_1 in the modalized version of R3 in Section 4.2 would require that no category in the secondary argument is marked with the unary modality '$\Box$', but β contains a category marked with '$\Box$'. Then the transformed derivation would be ungrammatical.

A second approach concerns the precise definition of the generalized composition rules, about which there is a surprising degree of disagreement. We have followed Hockenmaier and Young (2008) in classifying instances of generalized forward composition as harmonic if the innermost slash of the secondary argument is forward and crossed if it is backward. However, generalized forward composition is sometimes only accepted as harmonic if *all* slashes of the secondary argument are forward (see e.g. Baldridge (2002) (40, 41), Steedman (2001) (19)). At the same time, based on the principle that CCG rules should be derived from proofs of Categorial Type Logic as Baldridge (2002) does, it can be argued that generalized composition rules of the form $\mathsf{x/y}\ \ \mathsf{y/z\backslash w} \Rightarrow \mathsf{x/z\backslash w}$, which we have considered as harmonic, should actually be classified as crossed, due to the presence of a slash of opposite directionality in front of the w. This definition would break our proof. Thus our result might motivate further research on the 'correct' definition of generalized composition rules, which might then strengthen the generative capacity of pure MM-CCG.

5 Conclusion

In this paper, we have shown that the weak generative capacity of pure CCG and even pure B&K-CCG crucially depends on the ability to restrict the application of individual rules. This means that these formalisms cannot be fully lexicalized, in the sense that certain languages can only be described by selecting language-specific rules.

Our result generalizes Koller and Kuhlmann's (2009) result for pure *first-order* CCG. Our proof is not as different as it looks at first glance, as their construction of mapping a CCG derivation to a valency tree and back to a derivation provides a different transformation on derivation trees. Our transformation is also technically related to the normal form construction for CCG parsing presented by Eisner (1996).

Of course, at the end of the day, the issue that is more relevant to computational linguistics than a formalism's ability to generate artificial languages such as L_3 is how useful it is for modeling *natural* languages. CCG, and multi-modal CCG in particular, has a very good track record for this. In this sense, our formal result can also be understood as a contribution to a discussion about the expressive power that is needed to model natural languages.

Acknowledgments

We have profited enormously from discussions with Jason Baldridge and Mark Steedman, and would also like to thank the anonymous reviewers for their detailed comments.

References

Franz Baader and Tobias Nipkow. 1998. *Term Rewriting and All That*. Cambridge University Press.

Jason Baldridge and Geert-Jan M. Kruijff. 2003. Multi-modal Combinatory Categorial Grammar. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 211–218, Budapest, Hungary.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Yehoshua Bar-Hillel, Haim Gaifman, and Eli Shamir. 1964. On categorial and phrase structure grammars. In *Language and Information: Selected Essays on their Theory and Application*, pages 99–115. Addison-Wesley.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 176–182, Geneva, Switzerland.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

Haskell B. Curry, Robert Feys, and William Craig. 1958. *Combinatory Logic. Volume 1*. Studies in Logic and the Foundations of Mathematics. North-Holland.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 79–86, Santa Cruz, CA, USA.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 335–342, Philadelphia, USA.

Julia Hockenmaier and Peter Young. 2008. Non-local scrambling: the equivalence of TAG and CCG revisited. In *Proceedings of the 9th Internal Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, Tübingen, Germany.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer.

Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 460–468, Athens, Greece.

Michael Moortgat. 1997. Categorial type logics. In *Handbook of Logic and Language*, chapter 2, pages 93–177. Elsevier.

David Reitter, Julia Hockenmaier, and Frank Keller. 2006. Priming effects in combinatory categorial grammar. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 308–316, Sydney, Australia.

Mark Steedman and Jason Baldridge. 2010. Combinatory categorial grammar. In R. Borsley and K. Borjars, editors, *Non-Transformational Syntax*. Blackwell. Draft 7.0, to appear.

Mark Steedman. 2001. *The Syntactic Process*. MIT Press.

K. Vijay-Shanker and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

David J. Weir and Aravind K. Joshi. 1988. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 278–285, Buffalo, NY, USA.

Automatic Evaluation of Linguistic Quality in Multi-Document Summarization

Emily Pitler, Annie Louis, Ani Nenkova
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
epitler,lannie,nenkova@seas.upenn.edu

Abstract

To date, few attempts have been made to develop and validate methods for automatic evaluation of linguistic quality in text summarization. We present the first systematic assessment of several diverse classes of metrics designed to capture various aspects of well-written text. We train and test linguistic quality models on consecutive years of NIST evaluation data in order to show the generality of results. For grammaticality, the best results come from a set of syntactic features. Focus, coherence and referential clarity are best evaluated by a class of features measuring local coherence on the basis of cosine similarity between sentences, coreference information, and summarization specific features. Our best results are 90% accuracy for pairwise comparisons of competing systems over a test set of several inputs and 70% for ranking summaries of a specific input.

1 Introduction

Efforts for the development of automatic text summarizers have focused almost exclusively on improving content selection capabilities of systems, ignoring the linguistic quality of the system output. Part of the reason for this imbalance is the existence of ROUGE (Lin and Hovy, 2003; Lin, 2004), the system for automatic evaluation of content selection, which allows for frequent evaluation during system development and for reporting results of experiments performed outside of the annual NIST-led evaluations, the Document Understanding Conference (DUC)[1] and the Text Analysis Conference (TAC)[2]. Few metrics, however, have been proposed for evaluating linguistic quality and none have been validated on data from NIST evaluations.

In their pioneering work on automatic evaluation of summary coherence, Lapata and Barzilay (2005) provide a correlation analysis between human coherence assessments and (1) semantic relatedness between adjacent sentences and (2) measures that characterize how mentions of the same entity in different syntactic positions are spread across adjacent sentences. Several of their models exhibit a statistically significant agreement with human ratings and complement each other, yielding an even higher correlation when combined.

Lapata and Barzilay (2005) and Barzilay and Lapata (2008) both show the effectiveness of entity-based coherence in evaluating summaries. However, fewer than five automatic summarizers were used in these studies. Further, both sets of experiments perform evaluations of mixed sets of human-produced and machine-produced summaries, so the results may be influenced by the ease of discriminating between a human and machine written summary. Therefore, we believe it is an open question how well these features predict the quality of automatically generated summaries.

In this work, we focus on linguistic quality evaluation for *automatic systems only*. We analyze how well different types of features can rank good and poor machine-produced summaries. Good performance on this task is the most desired property of evaluation metrics during system development. We begin in Section 2 by reviewing the various aspects of linguistic quality that are relevant for machine-produced summaries and currently used in manual evaluations. In Section 3, we introduce and motivate diverse classes of features to capture vocabulary, sentence fluency, and local coherence properties of summaries. We evaluate the predictive power of these linguistic quality metrics by training and testing models on consecutive years of NIST evaluations (data described

[1] http://duc.nist.gov/
[2] http://www.nist.gov/tac/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 544–554,
Uppsala, Sweden, 11-16 July 2010.

in Section 4). We test the performance of different sets of features separately and in combination with each other (Section 5). Results are presented in Section 6, showing the robustness of each class and their abilities to reproduce human rankings of systems and summaries with high accuracy.

2 Aspects of linguistic quality

We focus on the five aspects of linguistic quality that were used to evaluate summaries in DUC: grammaticality, non-redundancy, referential clarity, focus, and structure/coherence.[3] For each of the questions, all summaries were manually rated on a scale from 1 to 5, in which 5 is the best.

The exact definitions that were provided to the human assessors are reproduced below.

Grammaticality: The summary should have no datelines, system-internal formatting, capitalization errors or obviously ungrammatical sentences (e.g., fragments, missing components) that make the text difficult to read.

Non-redundancy: There should be no unnecessary repetition in the summary. Unnecessary repetition might take the form of whole sentences that are repeated, or repeated facts, or the repeated use of a noun or noun phrase (e.g., "Bill Clinton") when a pronoun ("he") would suffice.

Referential clarity: It should be easy to identify who or what the pronouns and noun phrases in the summary are referring to. If a person or other entity is mentioned, it should be clear what their role in the story is. So, a reference would be unclear if an entity is referenced but its identity or relation to the story remains unclear.

Focus: The summary should have a focus; sentences should only contain information that is related to the rest of the summary.

Structure and Coherence: The summary should be well-structured and well-organized. The summary should not just be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

These five questions get at different aspects of what makes a well-written text. We therefore predict each aspect of linguistic quality separately.

3 Indicators of linguistic quality

Multiple factors influence the linguistic quality of text in general, including: word choice, the reference form of entities, and local coherence. We extract features which serve as proxies for each of the factors mentioned above (Sections 3.1 to 3.5). In addition, we investigate some models of grammaticality (Chae and Nenkova, 2009) and coherence (Graesser et al., 2004; Soricut and Marcu, 2006; Barzilay and Lapata, 2008) from prior work (Sections 3.6 to 3.9).

[3] `http://www-nlpir.nist.gov/projects/duc/duc2006/quality-questions.txt`

All of the features we investigate can be computed automatically directly from text, but some require considerable linguistic processing. Several of our features require a syntactic parse. To extract these, all summaries were parsed by the Stanford parser (Klein and Manning, 2003).

3.1 Word choice: language models

Psycholinguistic studies have shown that people read frequent words and phrases more quickly (Haberlandt and Graesser, 1985; Just and Carpenter, 1987), so the words that appear in a text might influence people's perception of its quality. Language models (LM) are a way of computing how familiar a text is to readers using the distribution of words from a large background corpus. Bigram and trigram LMs additionally capture grammaticality of sentences using properties of local transitions between words. For this reason, LMs are widely used in applications such as generation and machine translation to guide the production of sentences. Judging from the effectiveness of LMs in these applications, we expect that they will provide a strong baseline for the evaluation of at least some of the linguistic quality aspects.

We built unigram, bigram, and trigram language models with Good-Turing smoothing over the New York Times (NYT) section of the English Gigaword corpus (over 900 million words). We used the SRI Language Modeling Toolkit (Stolcke, 2002) for this purpose. For each of the three ngram language models, we include the *min*, *max*, and *average* log probability of the sentences contained in a summary, as well as the *overall log probability* of the entire summary.

3.2 Reference form: Named entities

This set of features examines whether named entities have informative descriptions in the summary. We focus on named entities because they appear often in summaries of news documents and are often not known to the reader beforehand. In addition, first mentions of entities in text introduce the entity into the discourse and so must be informative and properly descriptive (Prince, 1981; Fraurud, 1990; Elsner and Charniak, 2008).

We run the Stanford Named Entity Recognizer (Finkel et al., 2005) and record the number of *PERSONs*, *ORGANIZATIONs*, and *LOCATIONs*.

First mentions to people Feature exploration on our development set found that under-specified

references to people are much more disruptive to a summary than short references to organizations or locations. In fact, prior work in Nenkova and McKeown (2003) found that summaries that have been rewritten so that first mentions of people are informative descriptions and subsequent mentions are replaced with more concise reference forms are overwhelmingly preferred to summaries whose entity references have not been rewritten.

In this class, we include features that reflect the modification properties of noun phrases (NPs) in the summary that are first mentions to people. Noun phrases can include pre-modifiers, appositives, prepositional phrases, etc. Rather than prespecifying all the different ways a person expression can be modified, we hoped to discover the best patterns automatically, by including features for the average number of *each Part of Speech (POS) tag occurring before*, *each syntactic phrase occurring before*[4], *each POS tag occurring after*, and *each syntactic phrase occurring after* the head of the first mention NP for a PERSON. To measure if the lack of pre or post modification is particularly detrimental, we also include the proportion of PERSON first mention NPs *with no words before* and *with no words after* the head of the NP.

Summarization specific Most summarization systems today are *extractive* and create summaries using complete sentences from the source documents. A subsequent mention of an entity in a *source document* which is extracted to be the first mention of the entity in the *summary* is probably not informative enough. For each type of named entity (PERSON, ORGANIZATION, LOCATION), we separately record the number of instances which appear as first mentions in the summary but correspond to non-first mentions in the source documents.

3.3 Reference form: NP syntax

Some summaries might not include people and other named entities at all. To measure how entities are referred to more generally, we include features about the overall syntactic patterns found in NPs: the average number of *each POS tag* and *each syntactic phrase* occurring inside NPs.

[4]We define a linear order based on a preorder traversal of the tree, so syntactic phrases which dominate the head are considered occurring before the head.

3.4 Local coherence: Cohesive devices

In coherent text, constituent clauses and sentences are related and depend on each other for their interpretation. Referring expressions such as pronouns link the current utterance to those where the entities were previously mentioned. In addition, discourse connectives such as “but” or “because” relate propositions or events expressed by different clauses or sentences. Both these categories are known cohesive or linking devices in human-produced text (Halliday and Hasan, 1976). The mere presence of such items in a text would be indicative of better structure and coherence.

We compute a number of shallow features that provide a cheap way of capturing the above intuitions: the number of *demonstratives*, *pronouns*, and *definite descriptions* as well as the number of *sentence-initial discourse connectives*.

3.5 Local coherence: Continuity

This class of linguistic quality indicators is a combination of factors related to coreference, adjacent sentence similarity, and summary-specific context of surface cohesive devices.

Summarization specific Extractive multi-document summaries often lack appropriate antecedents for pronouns and proper context for the use of discourse connectives.

In fact, early work in summarization (Paice, 1980; Paice, 1990) has pointed out that the presence of cohesive devices described in the previous section might in fact be the source of problems. A manual analysis of automatic summaries (Otterbacher et al., 2002) also revealed that anaphoric references that cannot be resolved and unclear discourse relations constitute more than 30% of all revisions required to manually rewrite summaries into a more coherent form.

To identify these potential problems, we adapt the features for surface cohesive devices to indicate whether referring expressions and discourse connectives appear in the summary with the same context as in the input documents.

For each of the cohesive devices discussed in Section 3.4—*demonstratives*, *pronouns*, *definite descriptions*, and *sentence-initial discourse connectives*—we compare the previous sentence in the summary with the previous sentence in the input article. Two features are computed for each type of cohesive device: (1) number of times the preceding sentence in the summary is the same

as the preceding sentence in the input and (2) the number of times the preceding sentence in summary is different from that in the input. Since the previous sentence in the input text often contains the antecedent of pronouns in the current sentence, if the previous sentence from the input is also included in the summary, the pronoun is highly likely to have a proper antecedent.

We also compute the proportion of adjacent sentences in the summary that were extracted from the same input document.

Coreference Steinberger et al. (2007) compare the coreference chains in input documents and in summaries in order to locate potential problems. We instead define a set of more general features related to coreference that are not specific to summarization and are applicable for any text. Our features check the existence of proper antecedents for pronouns in the summary without reference to the text of the input documents.

We use the publicly available pronoun resolution system described in Charniak and Elsner (2009) to mark possible antecedents for pronouns in the summary. We then compute as features the number of times an antecedent for a pronoun was found *in the previous sentence*, *in the same sentence*, or *neither*. In addition, we modified the pronoun resolution system to also output the probability of the most likely antecedent and include the *average antecedent probability* for the pronouns in the text. Automatic coreference systems are trained on human-produced texts and we expect their accuracies to drop when applied to automatically generated summaries. However, the predictions and confidence scores still reflect whether or not possible antecedents exist in previous sentences that match in gender/number, and so may still be useful for coherence evaluation.

Cosine similarity We use cosine similarity to compute the overlap of words in adjacent sentences s_i and s_{i+1} as a measure of continuity.

$$cos\theta = \frac{v_{s_i}.v_{s_{i+1}}}{||v_{s_i}||||v_{s_{i+1}}||} \quad (1)$$

The dimensions of the two vectors (v_{s_i} and $v_{s_{i+1}}$) are the total number of word types from both sentences s_i and s_{i+1}. Stop words were retained. The value of each dimension for a sentence is the number of tokens of that word type in that sentence. We compute the *min*, *max*, and *average* value of cosine similarity over the entire summary.

While some repetition is beneficial for cohesion, too much repetition leads to redundancy in the summary. Cosine similarity is thus indicative of both continuity and redundancy.

3.6 Sentence fluency: Chae and Nenkova (2009)

We test the usefulness of a suite of 38 shallow syntactic features studied by Chae and Nenkova (2009). These features are weakly but significantly correlated with the fluency of machine translated sentences. These include *sentence length*, *number of fragments*, *average lengths of the different types of syntactic phrases*, *total length of modifiers in noun phrases*, and various other syntactic features. We expect that these structural features will be better at detecting ungrammatical sentences than the local language model features.

Since all of these features are calculated over individual sentences, we use the average value over all the sentences in a summary in our experiments.

3.7 Coh-Metrix: Graesser et al. (2004)

The Coh-Metrix tool[5] provides an implementation of 54 features known in the psycholinguistic literature to correlate with the coherence of human-written texts (Graesser et al., 2004). These include commonly used readability metrics based on sentence length and number of syllables in constituent words. Other measures implemented in the system are surface text properties known to contribute to text processing difficulty. Also included are measures of cohesion between adjacent sentences such as similarity under a latent semantic analysis (LSA) model (Deerwester et al., 1990), stem and content word overlap, syntactic similarity between adjacent sentences, and use of discourse connectives. Coh-Metrix has been designed with the goal of capturing properties of coherent text and has been used for grade level assessment, predicting student essay grades, and various other tasks. Given the heterogeneity of features in this class, we expect that they will provide reasonable accuracies for all the linguistic quality measures. In particular, the overlap features might serve as a measure of redundancy and local coherence.

[5] `http://cohmetrix.memphis.edu/`

3.8 Word coherence: Soricut and Marcu (2006)

Word co-occurrence patterns across adjacent sentences provide a way of measuring local coherence that is not linguistically informed but which can be easily computed using large amounts of unannotated text (Lapata, 2003; Soricut and Marcu, 2006). Word coherence can be considered as the analog of language models at the inter-sentence level. Specifically, we used the two features introduced by Soricut and Marcu (2006).

Soricut and Marcu (2006) make an analogy to machine translation: two words are likely to be translations of each other if they often appear in *parallel* sentences; in texts, two words are likely to signal local coherence if they often appear in *adjacent* sentences. The two features we computed are *forward likelihood*, the likelihood of observing the words in sentence s_i conditioned on s_{i-1}, and *backward likelihood*, the likelihood of observing the words in sentence s_i conditioned on sentence s_{i+1}. "Parallel texts" of 5 million adjacent sentences were extracted from the NYT section of GigaWord. We used the GIZA++[6] implementation of IBM Model 1 to align the words in adjacent sentences and obtain all relevant probabilities.

3.9 Entity coherence: Barzilay and Lapata (2008)

Linguistic theories, and Centering theory (Grosz et al., 1995) in particular, have hypothesized that the properties of the transition of attention from entities in one sentence to those in the next, play a major role in the determination of local coherence. Barzilay and Lapata (2008), inspired by Centering, proposed a method to compute the local coherence of texts on the basis of the sequences of entity mentions appearing in them.

In their Entity Grid model, a text is represented by a matrix with rows corresponding to each sentence in a text, and columns to each entity mentioned anywhere in the text. The value of a cell in the grid is the entity's grammatical role in that sentence (Subject, Object, Neither, or Absent). An entity transition is a particular entity's role in two adjacent sentences. The actual entity coherence features are the fraction of each type of these transitions in the entire entity grid for the text. One would expect that coherent texts would contain a certain distribution of entity transitions which would differ from those in incoherent sequences.

We use the Brown Coherence Toolkit[7] (Elsner et al., 2007) to construct the grids. The tool does not perform full coreference resolution. Instead, noun phrases are considered to refer to the same entity if their heads are identical.

Entity coherence features are the only ones that have been previously applied with success for predicting summary coherence. They can therefore be considered to be the state-of-the-art approach for automatic evaluation of linguistic quality.

4 Summarization data

For our experiments, we use data from the multi-document summarization tasks of the Document Understanding Conference (DUC) workshops (Over et al., 2007).

Our training and development data comes from DUC 2006 and our test data from DUC 2007. These were the most recent years in which the summaries were evaluated according to specific linguistic quality questions. Each input consists of a set of 25 related documents on a topic and the target length of summaries is 250 words.

In DUC 2006, there were 50 inputs to be summarized and 35 summarization systems which participated in the evaluation. This included 34 automatic systems submitted by participants, and a baseline system that simply extracted the leading sentences from the most recent article. In DUC 2007, there were 45 inputs and 32 different summarization systems. Apart from the leading sentences baseline, a high performance automatic summarizer from a previous year was also used as a baseline. All these automatic systems are included in our evaluation experiments.

4.1 System performance on linguistic quality

Each summary was evaluated according to the five linguistic quality questions introduced in Section 2: grammaticality, non-redundancy, referential clarity, focus, and structure. For each of these questions, all summaries were manually rated on a scale from 1 to 5, in which 5 is the best.

The distributions of system scores in the 2006 data are shown in Figure 1. Systems are currently the worst at structure, middling at referential clarity, and relatively better at grammaticality, focus,

[6] http://www.fjoch.com/GIZA++.html

[7] http://www.cs.brown.edu/~melsner/manual.html

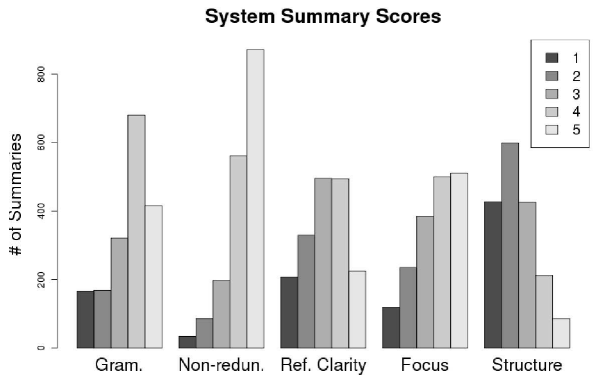


Figure 1: Distribution of system scores on the five linguistic quality questions

	Gram	Non-redun	Ref	Focus	Struct
Content	.02	-.40 *	.29	.28	.09
Gram		.38 *	.25	.24	.54 *
Non-redun			-.07	-.09	.27
Ref				.89 *	.76 *
Focus					.80 *

Table 1: Spearman correlations between the manual ratings for systems averaged over the 50 inputs in 2006; * $p < .05$

and non-redundancy. Structure is the aspect of linguistic quality where there is the most room for improvement. The only system with an average structure score above 3.5 in DUC 2006 was the leading sentences baseline system.

As can be expected, people are unlikely to be able to focus on a single aspect of linguistic quality exclusively while ignoring the rest. Some of the linguistic quality ratings are significantly correlated with each other, particularly referential clarity, focus, and structure (Table 1).

More importantly, the systems that produce summaries with good content[8] are not necessarily the systems producing the most readable summaries. Notice from the first row of Table 1 that none of the system rankings based on these measures of linguistic quality are significantly *positively* correlated with system rankings of content. The development of automatic linguistic quality measurements will allow researchers to optimize both content and linguistic quality.

[8]as measured by summary responsiveness ratings on a 1 to 5 scale, without regard to linguistic quality

5 Experimental setup

We use the summaries from DUC 2006 for training and feature development and DUC 2007 served as the test set. Validating the results on consecutive years of evaluation is important, as results that hold for the data in one year might not carry over to the next, as happened for example in Conroy and Dang (2008)'s work.

Following Barzilay and Lapata (2008), we report summary ranking accuracy as the fraction of correct pairwise rankings in the test set.

We use a Ranking SVM (SVM^{light} (Joachims, 2002)) to score summaries using our features. The Ranking SVM seeks to minimize the number of discordant pairs (pairs in which the gold standard has x_1 ranked strictly higher than x_2, but the learner ranks x_2 strictly higher than x_1). The output of the ranker is always a real valued score, so a global rank order is always obtained. The default regularization parameter was used.

5.1 Combining predictions

To combine information from the different feature classes, we train a meta ranker using the predictions from each class as features.

First, we use a leave-one out (jackknife) procedure to get the predictions of our features for the entire 2006 data set. To predict rankings of systems on one input, we train all the individual rankers, one for each of the classes of features introduced above, on data from the remaining inputs. We then apply these rankers to the summaries produced for the held-out input. By repeating this process for each input in turn, we obtain the predicted scores for each summary.

Once this is done, we use these predicted scores as features for the meta ranker, which is trained on all 2006 data. To test on a new summary pair in 2007, we first apply each individual ranker to get its predictions, and then apply the meta ranker.

In either case (meta ranker or individual feature class), all training is performed on 2006 data, and all testing is done on 2007 data which guarantees the results generalize well at least from one year of evaluation to the next.

5.2 Evaluation of rankings

We examine the predictive power of our features for each of the five linguistic quality questions in two settings. In *system-level* evaluation, we would like to rank all participating systems according to

their performance on the entire test set. In *input-level* evaluation, we would like to rank all summaries produced for a single given input.

For input-level evaluation, the pairs are formed from summaries of the *same input*. Pairs in which the gold standard ratings are tied are not included. After removing the ties, the test set consists of 13K to 16K pairs for each linguistic quality question. Note that there were 45 inputs and 32 automatic systems in DUC 2007. So, there are a total of $45 \cdot \binom{32}{2} = 22,320$ possible summary pairs.

For system-level evaluation, we treat the real-valued output of the SVM ranker for each summary as the linguistic quality score. The 45 individual scores for summaries produced by a given system are averaged to obtain an overall score for the system. The gold-standard system-level quality rating is equal to the *average human ratings* for the system's summaries over the 45 inputs. At the system level, there are about 500 non-tied pairs in the test set for each question.

For both evaluation settings, a random baseline which ranked the summaries in a random order would have an expected pairwise accuracy of 50%.

6 Results and discussion

6.1 System-level evaluation

System-level accuracies for each class of features are shown in Table 2. All classes of features perform well, with at least a 20% absolute increase in accuracy over the random baseline (50% accuracy). For each of the linguistic quality questions, the corresponding best class of features gives prediction accuracies around 90%. In other words, if these features were used to fully automatically compare systems that participated in the 2007 DUC evaluation, only one out of ten comparisons would have been incorrect. These results set a high standard for future work on automatic system-level evaluation of linguistic quality.

The state-of-the-art entity coherence features perform well but are not the best for any of the five aspects of linguistic quality. As expected, sentence fluency is the best feature class for grammaticality. For all four other questions, the best feature set is Continuity, which is a combination of summarization specific features, coreference features and cosine similarity of adjacent sentences. Continuity features outperform entity coherence by 3 to 4% absolute difference on referential quality, focus, and coherence. Accuracies from the language model features are within 1% of entity coherence for these three aspects of summary quality.

Feature set	Gram.	Redun.	Ref.	Focus	Struct.
Lang. models	87.6	83.0	91.2	85.2	86.3
Named ent.	78.5	83.6	82.1	74.0	69.6
NP syntax	85.0	83.8	87.0	76.6	79.2
Coh. devices	82.1	79.5	82.7	82.3	83.7
Continuity	88.8	**88.5**	**92.9**	**89.2**	**91.4**
Sent. fluency	**91.7**	78.9	87.6	82.3	84.9
Coh-Metrix	87.2	86.0	88.6	83.9	86.3
Word coh.	81.7	76.0	87.8	81.7	79.0
Entity coh.	90.2	88.1	89.6	85.0	87.1
Meta ranker	**92.9**	87.9	91.9	87.8	90.0

Table 2: System-level prediction accuracies (%)

Coh-Metrix, which has been proposed as a comprehensive characterization of text, does not perform as well as the language model and the entity coherence classes, which contain considerably fewer features related to only one aspect of text.

The classes of features specific to named entities and noun phrase syntax are the weakest predictors. It is apparent from the results that continuity, entity coherence, sentence fluency and language models are the most powerful classes of features that should be used in automation of evaluation and against which novel predictors of text quality should be compared.

Combining all feature classes with the meta ranker only yields higher results for grammaticality. For the other aspects of linguistic quality, it is better to use Continuity by itself to rank systems.

One certainly unexpected result is that features designed to capture one aspect of well-written text turn out to perform well for other questions as well. For instance, entity coherence and continuity features predict grammaticality with very high accuracy of around 90%, and are surpassed only by the sentence fluency features. These findings warrant further investigation because we would not expect characteristics of local transitions indicative of text structure to have anything to do with sentence grammaticality or fluency. The results are probably due to the significant correlation between structure and grammaticality (Table 1).

6.2 Input-level evaluation

The results of the input-level ranking experiments are shown in Table 3. Understandably, input-level prediction is more difficult and the results are lower compared to the system-level predictions: even with wrong predictions for some of the summaries by two systems, the overall judgment that

one system is better than the other over the entire test set can still be accurate.

While for system-level predictions the meta ranker was only useful for grammaticality, at the input level it outperforms every individual feature class for each of the five questions, obtaining accuracies around 70%.

These input-level accuracies compare favorably with automatic evaluation metrics for other natural language processing tasks. For example, at the 2008 ACL Workshop on Statistical Machine Translation, all fifteen automatic evaluation metrics, including variants of BLEU scores, achieved between 42% and 56% pairwise accuracy with human judgments at the sentence level (Callison-Burch et al., 2008).

As in system-level prediction, for referential clarity, focus, and structure, the best feature class is Continuity. Sentence fluency again is the best class for identifying grammaticality.

Coh-Metrix features are now best for determining redundancy. Both Coh-Metrix and Continuity (the top two features for redundancy) include overlap measures between adjacent sentences, which serve as a good proxy for redundancy.

Surprisingly, the *relative* performance of the feature classes at input level is not the same as for system-level prediction. For example, the language model features, which are the second best class for the system-level, do not fare as well at the input-level. Word co-occurrence which obtained good accuracies at the system level is the least useful class at the input level with accuracies just above chance in all cases.

6.3 Components of continuity

The class of features capturing sentence-to-sentence continuity in the summary (Section 3.5) are the most effective for predicting referential clarity, focus, and structure at the input level. We now investigate to what extent each of its components–summary-specific features, coreference, and cosine similarity between adjacent sentences–contribute to performance.

Results obtained after excluding each of the components of continuity is shown in Table 4; each line in the table represents Continuity minus a feature subclass. Removing cosine overlap causes the largest drop in prediction accuracy, with results about 10% lower than those for the complete Continuity class. Summary specific features, which compare the context of a sentence in the summary with the context in the original document where it appeared, also contribute substantially to the success of the Continuity class in predicting structure and referential clarity. Accuracies drop by about 7% when these features are excluded. However, the coreference features do not seem to contribute much towards predicting summary linguistic quality. The accuracies of the Continuity class are not affected at all when these coreference features are not included.

Feature set	Gram.	Redun.	Ref.	Focus	Struct.
Lang. models	66.3	57.6	62.2	60.5	62.5
Named ent.	52.9	54.4	60.0	54.1	52.5
NP Syntax	59.0	50.8	59.1	54.5	55.1
Coh. devices	56.8	54.4	55.2	52.7	53.6
Continuity	61.7	62.5	**69.7**	**65.4**	**70.4**
Sent. fluency	**69.4**	52.5	64.4	61.9	62.6
Coh-Metrix	65.5	**67.6**	67.9	63.0	62.4
Word coh.	54.7	55.5	53.3	53.2	53.7
Entity coh.	61.3	62.0	64.3	64.2	63.6
Meta ranker	**71.0**	**68.6**	**73.1**	**67.4**	**70.7**

Table 3: Input-level prediction accuracies (%)

6.4 Impact of summarization methods

In this paper, we have discussed an analysis of the outputs of current research systems. Almost all of these systems still use *extractive* methods. The summarization specific continuity features reward systems that include the necessary preceding context from the original document. These features have high prediction accuracies (Section 6.3) of *linguistic quality*, however note that the supporting context could often contain less important *content*. Therefore, there is a tension between strategies for optimizing linguistic quality and for optimizing content, which warrants the development of abstractive methods.

As the field moves towards more *abstractive* summaries, we expect to see differences in both a) summary linguistic quality and b) the features predictive of linguistic aspects.

As discussed in Section 4.1, systems are currently worst at structure/coherence. However, grammaticality will become more of an issue as systems use sentence compression (Knight and Marcu, 2002), reference rewriting (Nenkova and McKeown, 2003), and other techniques to produce their own sentences.

The number of discourse connectives is currently significantly *negatively* correlated with structure/coherence (Spearman correlation of r =

	Ref.	Focus	Struct.
Continuity	69.7	65.4	70.4
- Sum-specific	63.9	64.2	63.5
- Coref	70.1	65.2	70.6
- Cosine	60.2	56.6	60.7

Table 4: Ablation within the Continuity class; pairwise accuracy for input-level predictions (%)

-.06, p = .008 on DUC 2006 system summaries). This can be explained by the fact that they often lack proper context in an extractive summary. However, an *abstractive* system could plan a discourse structure and insert appropriate connectives (Saggion, 2009). In this case, we would expect the presence of discourse connectives to be a mark of a well-written summary.

6.5 Results on human-written abstracts

Since abstractive summaries would have markedly different properties from extracts, it would be interesting to know how well these sets of features would work for predicting the quality of machine-produced abstracts. However, since current systems are extractive, such a data set is not available.

Therefore we experiment on *human-written* abstracts to get an estimate of the expected performance of our features on abstractive system summaries. In both DUC 2006 and DUC 2007, ten NIST assessors wrote summaries for the various inputs. There are four human-written summaries for each input and these summaries were judged on the same five linguistic quality aspects as the machine-written summaries. We train on the human-written summaries from DUC 2006 and test on the human-written summaries from DUC 2007, using the same set-up as in Section 5.

These results are shown in Table 5. We only report results on the input level, as we are interested in distinguishing between the quality of the summaries, not the NIST assessors' writing skills.

Except for grammaticality, the prediction accuracies of the best feature classes for human abstracts are better than those at input level for machine extracts. This result is promising, as it shows that similar features for evaluating linguistic quality will be valid for abstractive summaries as well.

Note however that the relative performance of the feature sets changes between the machine and human results. While for the machines Continuity feature class is the best predictor of referential clarity, focus, and structure (Table 3), for humans, language models and sentence fluency are best for these three aspects of linguistic quality. A possible explanation for this difference could be that in system-produced extracts, incoherent organization influences human perception of linguistic quality to a great extent and so local coherence features turned out very predictive. But in human summaries, sentences are clearly well-organized and here, continuity features appear less useful. Sentence level fluency seems to be more predictive of the linguistic quality of these summaries.

Feature set	Gram.	Redun.	Ref.	Focus	Struct.
Lang. models	52.1	60.8	76.5	**71.9**	**78.4**
Named ent.	62.5	66.7	47.1	43.9	59.1
NP Syntax	**64.6**	49.0	43.1	49.1	58.0
Coh. devices	54.2	**68.6**	66.7	49.1	64.8
Continuity	54.2	49.0	62.7	61.4	71.6
Sent. fluency	54.2	64.7	**80.4**	**71.9**	72.7
Coh-Metrix	54.2	52.9	68.6	56.1	69.3
Word coh.	62.5	58.8	62.7	70.2	60.2
Entity coh.	45.8	49.0	54.9	52.6	56.8
Meta ranker	62.5	56.9	**80.4**	50.9	67.0

Table 5: Input-level prediction accuracies for human-written summaries (%)

7 Conclusion

We have presented an analysis of a wide variety of features for the linguistic quality of summaries. Continuity between adjacent sentences was consistently indicative of the quality of machine generated summaries. Sentence fluency was useful for identifying grammaticality. Language model and entity coherence features also performed well and should be considered in future endeavors for automatic linguistic quality evaluation.

The high prediction accuracies for input-level evaluation and the even higher accuracies for system-level evaluation confirm that questions regarding the linguistic quality of summaries can be answered reasonably using existing computational techniques. Automatic evaluation will make testing easier during system development and enable reporting results obtained outside of the cycles of NIST evaluation.

Acknowledgments

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship and NSF CAREER award 0953445. We would like to thank Bonnie Webber for productive discussions.

References

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz, and J. Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106.

J. Chae and A. Nenkova. 2009. Predicting the fluency of text with shallow structural features: case studies of machine translation and human-written text. In *Proceedings of EACL*, pages 139–147.

E. Charniak and M. Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of EACL*, pages 148–156.

J.M. Conroy and H.T. Dang. 2008. Mind the gap: dangers of divorcing evaluations of summary content from linguistic quality. In *Proceedings of COLING*, pages 145–152.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

M. Elsner and E. Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of ACL/HLT: Short Papers*, pages 41–44.

M. Elsner, J. Austerweil, and E. Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of NAACL/HLT*.

J.R. Finkel, T. Grenager, and C. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370.

K. Fraurud. 1990. Definiteness and the processing of noun phrases in natural discourse. *Journal of Semantics*, 7(4):395.

A.C. Graesser, D.S. McNamara, M.M. Louwerse, and Z. Cai. 2004. Coh-Metrix: Analysis of text on cohesion and language. *Behavior Research Methods Instruments and Computers*, 36(2):193–202.

B. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

K.F. Haberlandt and A.C. Graesser. 1985. Component processes in text comprehension and some of their interactions. *Journal of Experimental Psychology: General*, 114(3):357–374.

M.A.K. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman Group Ltd, London, U.K.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

M.A. Just and P.A. Carpenter. 1987. *The psychology of reading and language comprehension*. Allyn and Bacon Boston, MA.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

K. Knight and D. Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

M. Lapata and R. Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *International Joint Conference On Artificial Intelligence*, volume 19, page 1085.

M. Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of ACL*, pages 545–552.

C.Y. Lin and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of NAACL/HLT*, page 78.

C.Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

A. Nenkova and K. McKeown. 2003. References to named entities: a corpus study. In *Proceedings of HLT/NAACL 2003 (short paper)*.

J. Otterbacher, D. Radev, and A. Luo. 2002. Revisions that improve cohesion in multi-document summaries: a preliminary study. In *Proceedings of the Workshop on Automatic Summarization, ACL*.

P. Over, H. Dang, and D. Harman. 2007. Duc in context. *Information Processing Management*, 43(6):1506–1520.

C.D. Paice. 1980. The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases. In *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, pages 172–191.

C.D. Paice. 1990. Constructing literature abstracts by computer: Techniques and prospects. *Information Processing Management*, 26(1):171–186.

E.F. Prince. 1981. Toward a taxonomy of given-new information. *Radical pragmatics*, 223:255.

H. Saggion. 2009. A Classification Algorithm for Predicting the Structure of Summaries. *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, page 31.

R. Soricut and D. Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of ACL*.

J. Steinberger, M. Poesio, M.A. Kabadjov, and K. Jeek. 2007. Two uses of anaphora resolution in summarization. *Information Processing Management*, 43(6):1663–1680.

A. Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, volume 3.

Identifying Non-explicit Citing Sentences for Citation-based Summarization

Vahed Qazvinian
Department of EECS
University of Michigan
Ann Arbor, MI
vahed@umich.edu

Dragomir R. Radev
Department of EECS and
School of Information
University of Michigan
Ann Arbor, MI
radev@umich.edu

Abstract

Identifying background (context) information in scientific articles can help scholars understand major contributions in their research area more easily. In this paper, we propose a general framework based on probabilistic inference to extract such context information from scientific papers. We model the sentences in an article and their lexical similarities as a *Markov Random Field* tuned to detect the patterns that context data create, and employ a *Belief Propagation* mechanism to detect likely context sentences. We also address the problem of generating surveys of scientific papers. Our experiments show greater pyramid scores for surveys generated using such context information rather than citation sentences alone.

1 Introduction

In scientific literature, scholars use citations to refer to external sources. These secondary sources are essential in comprehending the new research. Previous work has shown the importance of citations in scientific domains and indicated that citations include survey-worthy information (Siddharthan and Teufel, 2007; Elkiss et al., 2008; Qazvinian and Radev, 2008; Mohammad et al., 2009; Mei and Zhai, 2008).

A citation to a paper in a scientific article may contain explicit information about the cited research. The following example is an excerpt from a CoNLL paper[1] that contains information about Eisner's work on bottom-up parsers and the notion of span in parsing:

"*Another use of bottom-up is due to* **Eisner (1996)**, *who introduced the notion of a span.*"

[1]Buchholz and Marsi "CoNLL-X Shared Task On Multilingual Dependency Parsing", CoNLL 2006

However, the citation to a paper may not always include explicit information about the cited paper:

"*This approach is one of those described in* **Eisner (1996)**"

Although this sentence alone does not provide any information about the cited paper, it suggests that its surrounding sentences describe the proposed approach in Eisner's paper:

> "*... In an all pairs approach, every possible pair of two tokens in a sentence is considered and some score is assigned to the possibility of this pair having a (directed) dependency relation. Using that information as building blocks, the parser then searches for the best parse for the sentence. This approach is one of those described in* **Eisner (1996)**."

We refer to such *implicit citations* that contain information about a specific secondary source but do not explicitly cite it, as sentences with *context* information or *context sentences* for short. We look at the patterns that such sentences create and observe that context sentences occur with-ing a small neighborhood of explicit citations. We also discuss the problem of extracting context sentences for a source-reference article pair. We propose a general framework that looks at each sentence as a random variable whose value determines its state about the target paper. In summary, our proposed model is based on the probabilistic inference of these random variables using graphical models. Finally we give evidence on how such sentences can help us produce better surveys of research areas. The rest of this paper is organized as follows. Preceded by a review of prior work in Section 2, we explain the data collection and our annotation process in Section 3. Section 4 explains our methodology and is followed by experimental setup in Section 5.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 555–564,
Uppsala, Sweden, 11-16 July 2010.

ACL-ID	Author	Title	Year	#Refs all	#Refs AAN	# Sents
P08-2026	McClosky & Charniak	Self-Training for Biomedical Parsing	2008	12	8	102
N07-1025*	Mihalcea	Using Wikipedia for Automatic ...	2007	21	12	153
N07-3002	Wang	Learning Structured Classifiers ...	2007	22	14	74
P06-1101	Snow et, al.	Semantic Taxonomy Induction ...	2006	19	9	138
P06-1116	Abdalla & Teufel	A Bootstrapping Approach To ...	2006	24	10	231
W06-2933	Nivre et, al.	Labeled Pseudo-Projective Dependency ...	2006	27	5	84
P05-1044	Smith & Eisner	Contrastive Estimation: Training Log-Linear ...	2005	30	13	262
P05-1073	Toutanova et, al.	Joint Learning Improves Semantic Role Labeling	2005	14	10	185
N03-1003	Barzilay & Lee	Learning To Paraphrase: An Unsupervised ...	2003	26	13	203
N03-2016*	Kondrak et, al.	Cognates Can Improve Statistical Translation ...	2003	8	5	92

Table 1: Papers chosen from AAN as source papers for the evaluation corpus, together with their publication year, number of references (in AAN) and number of sentences. Papers marked with $*$ are used to calculate inter-judge agreement.

2 Prior Work

Analyzing the structure of scientific articles and their relations has received a lot of attention recently. The structure of citation and collaboration networks has been studied in (Teufel et al., 2006; Newman, 2001), and summarization of scientific documents is discussed in (Teufel and Moens, 2002). In addition, there is some previous work on the importance of citation sentences. Elkiss et al, (Elkiss et al., 2008) perform a large-scale study on citations in the free PubMed Central (PMC) and show that they contain information that may not be present in abstracts. In other work, Nanba et al, (Nanba and Okumura, 1999; Nanba et al., 2004b; Nanba et al., 2004a) analyze citation sentences and automatically categorize them in order to build a tool for survey generation.

The text of scientific citations has been used in previous research. Bradshaw (Bradshaw, 2002; Bradshaw, 2003) uses citations to determine the content of articles. Similarly, the text of citation sentences has been directly used to produce summaries of scientific papers in (Qazvinian and Radev, 2008; Mei and Zhai, 2008; Mohammad et al., 2009). Determining the scientific attribution of an article has also been studied before. Siddharthan and Teufel (Siddharthan and Teufel, 2007; Teufel, 2005) categorize sentences according to their role in the author's argument into predefined classes: Own, Other, Background, Textual, Aim, Basis, Contrast.

Little work has been done on automatic citation extraction from research papers. Kaplan et al, (Kaplan et al., 2009) introduces "citation-site" as a block of text in which the cited text is discussed. The mentioned work uses a machine learning method for extracting citations from research papers and evaluates the result using 4 annotated articles.

In our work we use graphical models to extract context sentences. Graphical models have a number of properties and corresponding techniques and have been used before on Information Retrieval tasks. Romanello et al, (Romanello et al., 2009) use Conditional Random Fields (CRF) to extract references from unstructured text in digital libraries of classic texts. Similar work include term dependency extraction (Metzler and Croft, 2005), query expansion (Metzler and Croft, 2007), and automatic feature selection (Metzler, 2007).

3 Data

The ACL Anthology Network (AAN)[2] is a collection of papers from the ACL Anthology[3] published in the Computational Linguistics journal and proceedings from ACL conferences and workshops and includes more than $14,000$ papers over a period of four decades (Radev et al., 2009). AAN includes the citation network of the papers in the ACL Anthology. The papers in AAN are publicly available in text format retrieved by an OCR process from the original pdf files, and are segmented into sentences.

To build a corpus for our experiments we picked 10 recently published papers from various areas in NLP[4], each of which had references for a total of 203 candidate paper-reference pairs. Table 1 lists these papers together with their authors, titles, publication year, number of references, number of references within AAN, and the number of sen-

[2]http://clair.si.umich.edu/clair/anthology/

[3]http://www.aclweb.org/anthology-new/

[4]Regardless of data selection, the methodology in this work is applicable to any of the papers in AAN.

L&P	S&al	Sentence
		. . .
$\mathcal{C}$	$\mathcal{C}$	Jacquemin (1999) and Barzilay and McKeown (2001) identify phrase level paraphrases, while **Lin and Pantel (2001)** and **Shinyama et al. (2002)** acquire structural paraphrases encoded as templates.
1	1	These latter are the most closely related to the sentence-level paraphrases we desire, and so we focus in this section on template-induction approaches.
$\mathcal{C}$	0	**Lin and Pantel (2001)** extract inference rules, which are related to paraphrases (for example, X wrote Y implies X is the author of Y), to improve question answering.
1	0	They assume that paths in dependency trees that take similar arguments (leaves) are close in meaning.
1	0	However, only two-argument templates are considered.
0	$\mathcal{C}$	**Shinyama et al. (2002)** also use dependency-tree information to extract templates of a limited form (in their case, determined by the underlying information extraction application).
1	1	Like us (and unlike Lin and Pantel, who employ a single large corpus), they use articles written about the same event in different newspapers as data.
1	1	Our approach shares two characteristics with the two methods just described: pattern comparison by analysis of the patterns respective arguments, and use of nonparallel corpora as a data source.
0	0	However, extraction methods are not easily extended to generation methods.
1	1	One problem is that their templates often only match small fragments of a sentence.
1	1	While this is appropriate for other applications, deciding whether to use a given template to generate a paraphrase requires information about the surrounding context provided by the entire sentence.
		. . .

Table 2: Part of the annotation for N03-1003 with respect to two of its references "Lin and Pantel (2001)" (the first column) "Shinyama et al. (2002)" (the second column). $\mathcal{C}$s indicate explicit citations, 1s indicate implicit citations and 0s are none.

tences.

3.1 Annotation Process

We annotated the sentences in each paper from Table 1. Each *annotation instance* in our setting corresponds to a paper-reference pair, and is a vector in which each dimension corresponds to a sentence and is marked with a $\mathcal{C}$ if it explicitly cites the reference, and with a 1 if it implicitly talks about it. All other sentences are marked with 0s. Table 2 shows a portion of two separate annotation instances of N03-1003 corresponding to two of its references. Our annotation has resulted in 203 annotation instances each corresponding to one paper-reference pair. The goal of this work is to automatically identify all context sentences, which are marked as "1".

3.1.1 Inter-judge Agreement

We also asked a neutral annotator[5] to annotate two of our datasets that are marked with $*$ in Table 1. For each paper-reference pair, the annotator was provided with a vector in which explicit citations were already marked with $\mathcal{C}$s. The annotation guidelines instructed the annotator to look at each explicit citation sentence, and read up to 15 sentences before and after, then mark context sentences around that sentence with 1s. Next, the 29 annotation instances done by the external annotator were compared with the corresponding annotations that we did, and the Kappa coefficient (κ) was calculated. The κ statistic is formulated as

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

where $Pr(a)$ is the relative observed agreement among raters, and $Pr(e)$ is the probability that annotators agree by chance if each annotator is randomly assigning categories. To calculate κ, we ignored all explicit citations (since they were provided to the external annotator) and used the binary categories (i.e., 1 for context sentences, and 0 otherwise) for all other sentences. Table 3 shows the annotation vector size (i.e., number of sentences), number of annotation instances (i.e., number of references), and average κ for each set. The average κ is above 0.85 in both cases, suggesting that the annotation process has a low degree of subjectivity and can be considered reliable.

ACL-ID	vector size	# Annotations	$\overline{\kappa}$
N07-1025*	153	21	0.889 ± 0.30
N03-2016*	92	8	0.853 ± 0.35

Table 3: Average κ coefficient as inter-judge agreement for annotations of two sets

3.2 Analysis

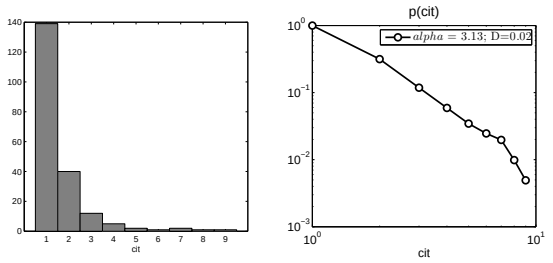
In this section we describe our analysis. First, we look at the number of explicit citations each reference has received in a paper. Figure 1 (a) shows the histogram corresponding to this distribution. It indicates that the majority of references get cited in only 1 sentence in a scientific article, while the maximum being 9 in our collected dataset with only 1 instance (i.e., there is only 1 reference that gets cited 9 times in a paper). Moreover, the data exhibits a highly positive-skewed distribution. This is illustrated on a log-log scale in Figure 1 (b). This highly skewed distribution indicates that the majority of references get cited only once in a citing paper. The very small number of citing sentences can not make a full inventory of the contributions of the cited paper, and therefore, extracting explicit citations alone without context

[5] Someone not involved with the paper but an expert in NLP.

gap size	0	1	2	4	9	10	15	16
instance	273	14	2	1	2	1	1	1

Table 4: The distribution of gaps in the annotated data

sentences may result in information loss about the contributions of the cited paper.



a b

Figure 1: (a) Histogram of the number of different citations to each reference in a paper. (b) The distribution observed for the number of different citations on a log-log scale.

Next, we investigate the distance between context sentences and the closest citations. For each context sentence, we find its distance to the closets context sentence or explicit citation. Formally, we define the *gap* to be the number of sentences between a context sentence (marked with 1) and the closest context sentence or explicit citation (marked with either C or 1) to it. For example, the second column of Table 2 shows that there is a gap of size 1 in the 9^{th} sentence in the set of context and citation sentences about Shinyama et al. (2002). Table 4 shows the distribution of gap sizes in the annotated data. This observation suggests that the majority of context sentences directly occur after or before a citation or another context sentence. However, it shows that gaps between sentences describing a cited paper actually exist, and a proposed method should have the capability to capture them.

4 Proposed Method

In this section we propose our methodology that enables us to identify the context information of a cited paper. Particularly, the task is to assign a binary label X_C to each sentence S_i from a paper S, where $X_C = 1$ shows a context sentence related to a given cited paper, C. To solve this problem we propose a systematic way to model the network level relationship between consecutive sentences. In summary, each sentence is represented with a node and is given two scores (context, non-context), and we update these scores to be in harmony with the neighbors' scores.

A particular class of graphical models known as *Markov Random Fields* (MRFs) are suited for solving inference problems with uncertainty in observed data. The data is modeled as an undirected graph with two types of nodes: hidden and observed. Observed nodes represent values that are known from the data. Each hidden node x_u, corresponding to an observed node y_u, represents the true state underlying the observed value. The state of a hidden node is related to the value of its corresponding observed node as well as the states of its neighboring hidden nodes.

The *local Markov property* of an MRF indicates that a variable is conditionally independent on all other variables given its neighbors: $x_v \perp\!\!\!\perp x_{V \setminus cl(v)} | x_{ne(v)}$, where $ne(v)$ is the set of neighbors of v, and $cl(v) = \{v\} \cup ne(v)$ is the closed neighborhood of v. Thus, the state of a node is assumed to statistically depend only upon its hidden node and each of its neighbors, and independent of any other node in the graph given its neighbors.

Dependencies in an MRF are represented using two functions: *Compatibility function (ψ)* and *Potential function (ϕ)*. $\psi_{uv}(x_c, x_d)$ shows the edge potential of an edge between two nodes u, v of classes x_c and x_d. Large values of ψ_{uv} would indicate a strong association between x_c and x_d at nodes u, v. The Potential function, $\phi_i(x_c, y_c)$, shows the statistical dependency between x_c and y_c at each node i assumed by the MRF model.

In order to find the marginal probabilities of x_is in a MRF we can use *Belief Propagation (BP)* (Yedidia et al., 2003). If we assume the y_is are fixed and show $\phi_i(x_i, y_i)$ by $\phi_i(x_i)$, we can find the joint probability distribution for unknown variables x_i as

$$p(\{x\}) = \frac{1}{Z} \prod_{ij} \psi_{ij}(x_i, x_j) \prod_i \phi_i(x_i)$$

In the BP algorithm a set of new variables m is introduced where $m_{ij}(x_j)$ is the message passed from i to j about what state x_j should be in. Each message, $m_{ij}(x_j)$, is a vector with the same dimensionality of x_j in which each dimension shows i's opinion about j being in the corresponding class. Therefore each message could be considered as a probability distribution and its components should sum up to 1. The final belief at a

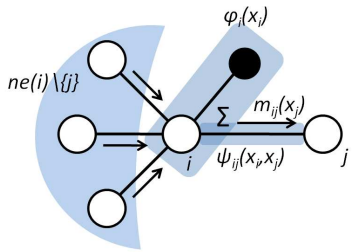


Figure 2: The illustration of the message updating rule. Elements that make up the message from a node i to another node j: messages from i's neighbors, local evidence at i, and propagation function between i, j summed over all possible states of node i.

node i, in the BP algorithm, is also a vector with the same dimensionality of messages, and is proportional to the local evidence as well as all messages from the node's neighbors:

$$b_i(x_i) \leftarrow k\phi_i(x_i) \prod_{j \in ne(i)} m_{ji}(x_i) \quad (1)$$

where k is the normalization factor of the beliefs about different classes. The message passed from i to j is proportional to the propagation function between i, j, the local evidence at i, and all messages sent to i from its neighbors except j:

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \in ne(i)\backslash j} m_{ki}(x_i) \quad (2)$$

Figure 2 illustrates the message update rule.

Convergence can be determined based on a variety of criteria. It can occur when the maximum change of any message between iteration steps is less than some threshold. Convergence is guaranteed for trees but not for general graphs. However, it typically occurs in practice (McGlohon et al., 2009). Upon convergence, belief scores are determined by Equation 1.

4.1 MRF construction

To find the sentences from a paper that form the context information of a given cited paper, we build an MRF in which a hidden node x_i and an observed node y_i correspond to each sentence. The structure of the graph associated with the MRF is dependent upon the validity of a basic assumption. This assumption indicates that the generation of a sentence (in form of its words) only depends on its surrounding sentences. Said differently, each sentence is written independently of all other sentences given a number of its neighbors. This local dependence assumption can result in a number of different MRFs, each built assuming a dependency between a sentence and all sentences within a particular distance. Figure 3 shows the structure of the two MRFs at either extreme of the local dependence assumption. In Figure 3 a, each sentence only depends on one following and one preceding sentence, while Figure 3 b shows an MRF in which sentences are dependent on each other regardless of their position. We refer to the former by $\mathbf{BP_1}$, and to the latter by $\mathbf{BP_n}$. Generally, we use $\mathbf{BP_i}$ to denote an MRF in which each sentence is connected to i sentences before and after.

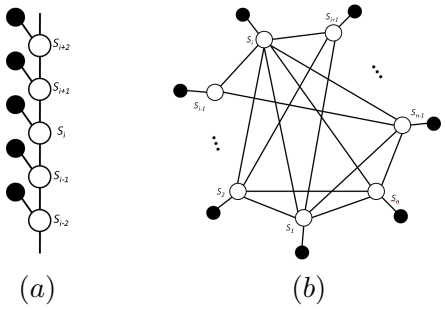


Figure 3: The structure of the MRF constructed based on the independence of non-adjacent sentences; (a) left, each sentence is independent on all other sentences given its immediate neighbors. (b) right, sentences have dependency relationship with each other regardless of their position.

$\psi_{ij}(x_c, x_d)$	$x_d = 0$	$x_d = 1$
$x_c = 0$	0.5	0.5
$x_c = 1$	$1 - S_{ij}$	S_{ij}

Table 5: The compatibility function ψ between any two nodes in the MRFs from the sentences in scientific papers

4.2 Compatibility Function

The compatibility function of an MRF represents the association between the hidden node classes. A node's belief to be in class 1 is its probability to be included in the context. The belief of a node i, about its neighbor j to be in either classes is assumed to be 0.5 if i is in class 0. In other words, if a node is not part of the context itself, we assume

it has no effect on its neighbors' classes. In contrast, if i is in class 1 its belief about its neighbor j is determined by their mutual lexical similarity. If this similarity is close to 1 it indicates a stronger tie between i, j. However, if i, j are not similar, i's probability of being in class 1, should not affect that of j's. To formalize this assumption we use the sigmoid of the cosine similarity of two sentences to build ψ. More formally, we define S to be

$$S_{ij} = \frac{1}{1 + e^{-cosine(i,j)}}$$

The sigmoid function obtains a value of 0.5 for a cosine of 0 indicating that there is no bias in the association of the two sentences. The matrix in Table 5 shows the compatibility function built based on the above arguments.

4.3 Potential Function

The node potential function of an MRF can incorporate some other features observable from data. Here, the goal is to find all sentences that are about a specific cited paper, without having explicit citations. To build the node potential function of the observed nodes, we use some sentence level features. First, we use the explicit citation as an important feature of a sentence. This feature can affect the belief of the corresponding hidden node, which can in turn affect its neighbors' beliefs. For a given paper-reference pair, we flag (with a 1) each sentence that has an explicit citation to the reference.

The second set of features that we are interested in are discourse-based features. In particular we match each sentence with specific patterns and flag those that match. The first pattern is a bigram in which the first term matches any of "*this; that; those; these; his; her; their; such; previous*", and the second term matches any of "*work; approach; system; method; technique; result; example*". The second pattern includes all sentences that start with "*this; such*".

Finally, the similarity of each sentence to the reference is observable from the data and can be used as a sentence-level feature. Intuitively, if a sentence has higher similarity with the reference paper, it should have a higher potential of being in class 1 or $\mathcal{C}$. The flag of each sentence here is a value between 0 and 1 and is determined by its cosine similarity to the reference. Once the flags for each sentence, S_i are determined, we calculate normalized f_i as the unweighted linear combination of individual features. Based on f_is, we compute the potential function, ϕ, as shown in Table 6.

$\phi_i(x_c, y_c)$	$x_c = 0$	$x_c = 1$
	$1 - f_i$	f_i

Table 6: The node potential function ϕ for each node in the MRFs from the sentences in scientific papers is built using the sentences' flags computed using sentence level features.

5 Experiments

The intrinsic evaluation of our methodology means to directly compare the output of our method with the gold standards obtained from the annotated data. Our methodology finds the sentences that cite a reference implicitly. Therefore the output of the inference method is a vector, υ, of 1's and 0's, whereby a 1 at element i means that sentence i in the source document is a context sentence about the reference while a 0 means an explicit citation or neither. The gold standard for each paper-reference pair, ω (obtained from the annotated vectors in Section 3.1 by changing all $\mathcal{C}$s to 0s), is also a vector of the same format and dimensionality.

Precision, recall, and F_β for this task can be defined as

$$p = \frac{\upsilon \cdot \omega}{\upsilon \cdot \mathbf{1}}; \quad r = \frac{\upsilon \cdot \omega}{\omega \cdot \mathbf{1}}; \quad F_\beta = \frac{(1 + \beta^2) p \cdot r}{\beta^2 p + r} \qquad (3)$$

where $\mathbf{1}$ is a vector of 1's with the same dimensionality and β is a non-negative real number.

5.1 Baseline Methods

The first baseline that we use is an IR-based method. This baseline, $\mathbf{B_1}$, takes explicit citations as an input but use them to find context sentences. Given a paper-reference pair, for each explicit citation sentence, marked with $\mathcal{C}$, $\mathbf{B_1}$ picks its preceding and following sentences if their similarities to that sentence is greater than a cutoff (the median of all such similarities), and repeats this for neighboring sentences of newly marked sentences. Intuitively, $\mathbf{B_1}$ tries to find the best chain (window) around citing sentences.

As the second baseline, we use the hand-crafted discourse based features used in MRF's potential function. Particularly, this baseline, $\mathbf{B_2}$, marks

paper	$\mathbf{B_1}$	$\mathbf{B_2}$	**SVM**	$\mathbf{BP_1}$	$\mathbf{BP_4}$	$\mathbf{BP_n}$
P08-2026	0.441	0.237	0.249	0.470	**0.613**	0.285
N07-1025	0.388	0.102	0.124	0.313	**0.466**	0.138
N07-3002	0.521	0.339	0.232	**0.742**	0.627	0.315
P06-1101	0.125	0.388	0.127	0.649	**0.889**	0.193
P06-1116	0.283	0.104	0.100	0.307	**0.341**	0.130
W06-2933	0.313	0.100	0.176	0.338	**0.413**	0.160
P05-1044	0.225	0.100	0.060	0.172	**0.586**	0.094
P05-1073	0.144	0.100	0.144	0.433	**0.518**	0.171
N03-1003	0.245	0.249	0.126	**0.523**	0.466	0.125
N03-2016	0.100	0.181	0.224	0.439	**0.482**	0.185

Table 7: Average $F_{\beta=3}$ for similarity based baseline ($\mathbf{B_1}$), discourse-based baseline ($\mathbf{B_2}$), a supervised method (**SVM**) and three MRF-based methods.

each sentence that is within a particular distance (4 in our experiments) of an explicit citation and matches one of the two patterns mentioned in Section 4.3. After marking all such sentences, $\mathbf{B_2}$ also marks all sentences between them and the closest explicit citation, which is no farther than 4 sentences away. This baseline helps us understand how effectively this sentence level feature can work in the absence of other features and the network structure.

Finally, we use a supervised method, **SVM**, to classify sentences as context/non-context. We use 4 features to train the **SVM** model. These 4 features comprise the 3 sentence level features used in MRF's potential function (i.e., similarity to reference, explicit citation, matching certain regular-expressions) and a network level feature: distance to the closes explicit citation. For each source paper, P, we use all other source papers and their source-reference annotation instances to train a model. We then use this model to classify all instances in P. Although the number of references and thus source-reference pairs are different for different papers, this can be considered similar to a 10-fold cross validation scheme, since for each source paper the model is built using all source-reference pairs of all other 9 papers.

We compare these baselines with 3 MRF-based systems each with a different assumption about independence of sentences. $\mathbf{BP_1}$ denotes an MRF in which each sentence is only connected to 1 sentence before and after. In $\mathbf{BP_4}$ locality is more relaxed and each sentence is connected to 4 sentences on each sides. $\mathbf{BP_n}$ denotes an MRF in which all sentences are connected to each other regardless of their position in the paper.

Table 7 shows $F_{\beta=3}$ for our experiments and shows how $\mathbf{BP_4}$ outperforms the other methods on average. The value 4 may suggest the fact that although sentences might be independent of distant sentences, they depend on more than one sentence on each side.

The final experiment we do to intrinsically evaluate the MRF-base method is to compare different sentence-level features. The first feature used to build the potential function is explicit citations. This feature does not directly affect context sentences (i.e., it affects the marginal probability of context sentences through the MRF network connections). Therefore, we do not alter this feature in comparing different features. However, we look at the effect of the second and the third features: hand-crafted regular expression-based features and similarity to the reference. For each paper, we use $\mathbf{BP_4}$ to perform 3 experiments: two in absence of each feature and one including all features. Figure 4 shows the average $F_{\beta=3}$ for each experiment. This plot shows that the features lead to better results when used together.

6 Impact on Survey Generation

We also performed an extrinsic evaluation of our context extraction methodology. Here we show how context sentences add important survey-worthy information to explicit citations. Previous work that generate surveys of scientific topics use the text of citation sentences alone (Mohammad et al., 2009; Qazvinian and Radev, 2008). Here, we show how the surveys generated using citations and their context sentences are better than those generated using citation sentences alone.

We use the data from (Mohammad et al., 2009)

... Naturally, our current work on question answering for the reading comprehension task is most related to those of (Hirschman et al. , 1999; Charniak et al. , 2000; Riloffand Thelen, 2000 ; Wang et al. , 2000). **In fact, all of this body of work as well as ours are evaluated on the same set of test stories, and are developed (or trained) on the same development set of stories.** The work of (Hirschman et al. , 1999) initiated this series of work, and it reported an accuracy of 36.3% on answering the questions in the test stories. **Subsequently, the work of (Riloffand Thelen , 2000) and (Chaxniak et al. , 2000) improved the accuracy further to 39.7% and 41%, respectively. However, all of these three systems used handcrafted, deterministic rules and algorithms**...
...**The cross-model comparison showed that the performance ranking of these models was: U-SVM** $>$ **PatternM** $>$ **S-SVM** $>$ **Retrieval-M.** Compared with retrieval-based [Yang et al. 2003], pattern-based [Ravichandran et al. 2002 and Soubbotin et al. 2002], and deep NLP-based [Moldovan et al. 2002, Hovy et al. 2001; and Pasca et al. 2001] answer selection, machine learning techniques are more effective in constructing QA components from scratch. **These techniques suffer, however, from the problem of requiring an adequate number of handtagged question-answer training pairs. It is too expensive and labor intensive to collect such training pairs for supervised machine learning techniques** ...
... **As expected, the definition and person-bio answer types are covered well by these resources.** The web has been employed for pattern acquisition (Ravichandran et al. , 2003), document retrieval (Dumais et al. , 2002), query expansion (Yang et al. , 2003), structured information extraction, and answer validation (Magnini et al. , 2002). **Some of these approaches enhance existing QA systems, while others simplify the question answering task, allowing a less complex approach to find correct answers** ...

Table 8: A portion of the QA survey generated by LexRank using the context information.

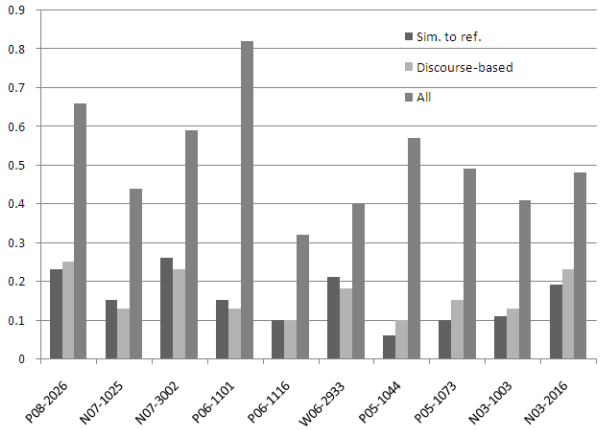


Figure 4: Average $F_{\beta=3}$ for $\mathbf{BP_4}$ employing different features.

	citation survey	context survey
	QA	
CT nuggets	0.416	**0.634**
AB nuggets	0.397	**0.594**
	DP	
CT nuggets	0.324	**0.379**

Table 9: Pyramid $F_{\beta=3}$ scores of automatic surveys of QA and DP data. The QA surveys are evaluated using nuggets drawn from citation texts (CT), or abstracts (AB), and DP surveys are evaluated using nuggets from citation texts (CT).

that contains two sets of cited papers and corresponding citing sentences, one on Question Answering (QA) with 10 papers and the other on Dependency Parsing (DP) with 16 papers. The QA set contains two different sets of nuggets extracted by experts respectively from paper abstracts and citation sentences. The DP set includes nuggets extracted only from citation sentences. We use these nugget sets, which are provided in form of regular expressions, to evaluate automatically generated summaries. To perform this experiment we needed to build a new corpus that includes context sentences. For each citation sentence, $\mathbf{BP_4}$ is used on the citing paper to extract the proper context. Here, we limit the context size to be 4 on each side. That is, we attach to a citing sentence any of its 4 preceding and following sentences if BP4 marks them as context sentences. Therefore, we build a new corpus in which each explicit citation sentence is replaced with the same sentence attached to at most 4 sentence on each side.

After building the context corpus, we use LexRank (Erkan and Radev, 2004) to generate 2 QA and 2 DP surveys using the citation sentences only, and the new context corpus explained above. LexRank is a multidocument summarization system, which first builds a cosine similarity graph of all the candidate sentences. Once the network is built, the system finds the most central sentences by performing a random walk on the graph. We limit these surveys to be of a maximum length of 1000 words. Table 8 shows a portion of the survey generated from the QA context corpus. This example shows how context sentences add meaningful and survey-worthy information along with citation sentences. Table 9 shows the Pyramid $F_{\beta=3}$ score of automatic surveys of QA and DP

data. The QA surveys are evaluated using nuggets drawn from citation texts (CT), or abstracts (AB), and DP surveys are evaluated using nuggets from citation texts (CT). In all evaluation instances the surveys generated with the context corpora excel at covering nuggets drawn from abstracts or citation sentences.

7 Conclusion

In this paper we proposed a framework based on probabilistic inference to extract sentences that appear in the scientific literature, and which are about a secondary source, but which do not contain explicit citations to that secondary source. Our methodology is based on inference in an MRF built using the similarity of sentences and their lexical features. We show, by numerical experiments, that an MRF in which each sentence is connected to only a few adjacent sentences properly fits this problem. We also investigate the usefulness of such sentences in generating surveys of scientific literature. Our experiments on generating surveys for Question Answering and Dependency Parsing show how surveys generated using such context information along with citation sentences have higher quality than those built using citations alone.

Generating fluent scientific surveys is difficult in absence of sufficient background information. Our future goal is to combine summarization and bibliometric techniques towards building automatic surveys that employ context information as an important part of the generated surveys.

8 Acknowledgments

The authors would like to thank Arzucan Özgür from University of Michigan for annotations.

This paper is based upon work supported by the National Science Foundation grant "iOPENER: A Flexible Framework to Support Rapid Learning in Unfamiliar Research Domains", jointly awarded to University of Michigan and University of Maryland as IIS 0705832. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Shannon Bradshaw. 2002. *Reference Directed Indexing: Indexing Scientific Literature in the Context of Its Use*. Ph.D. thesis, Northwestern University.

Shannon Bradshaw. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir R. Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*.

Dain Kaplan, Ryu Iida, and Takenobu Tokunaga. 2009. Automatic extraction of citation contexts for research paper summarization: A coreference-chain based approach. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 88–95, Suntec City, Singapore, August. Association for Computational Linguistics.

Mary McGlohon, Stephen Bay, Markus G. Anderle, David M. Steier, and Christos Faloutsos. 2009. Snare: a link analytic system for graph labeling and risk detection. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1265–1274.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL '08*, pages 816–824.

Donald Metzler and W. Bruce Croft. 2005. A markov random field model for term dependencies. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479.

Donald Metzler and W. Bruce Croft. 2007. Latent concept expansion using markov random fields. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 311–318.

Donald A. Metzler. 2007. Automatic feature selection in the markov random field model for information retrieval. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 253–262.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms.

In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 584–592, Boulder, Colorado, June. Association for Computational Linguistics.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *IJCAI1999*, pages 926–931.

Hidetsugu Nanba, Takeshi Abekawa, Manabu Okumura, and Suguru Saito. 2004a. Bilingual presri: Integration of multiple research paper databases. In *Proceedings of RIAO 2004*, pages 195–211, Avignon, France.

Hidetsugu Nanba, Noriko Kando, and Manabu Okumura. 2004b. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *Proceedings of the 11th SIG Classification Research Workshop*, pages 117–134, Chicago, USA.

Mark E. J. Newman. 2001. The structure of scientific collaboration networks. *PNAS*, 98(2):404–409.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008*, Manchester, UK.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *ACL workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

Matteo Romanello, Federico Boschetti, and Gregory Crane. 2009. Citations in the digital library of classics: Extracting canonical references by using conditional random fields. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 80–87, Suntec City, Singapore, August. Association for Computational Linguistics.

Advaith Siddharthan and Simone Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *Proceedings of NAACL/HLT-07*.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the EMNLP*, Sydney, Australia, July.

Simone Teufel. 2005. Argumentative Zoning for Improved Citation Indexing. *Computing Attitude and Affect in Text: Theory and Applications*, pages 159–170.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. pages 239–269.

Automatic Generation of Story Highlights

Kristian Woodsend and **Mirella Lapata**
School of Informatics, University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
k.woodsend@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we present a joint content selection and compression model for single-document summarization. The model operates over a phrase-based representation of the source document which we obtain by merging information from PCFG parse trees and dependency graphs. Using an integer linear programming formulation, the model learns to select and combine phrases subject to length, coverage and grammar constraints. We evaluate the approach on the task of generating "story highlights"—a small number of brief, self-contained sentences that allow readers to quickly gather information on news stories. Experimental results show that the model's output is comparable to human-written highlights in terms of both grammaticality and content.

1 Introduction

Summarization is the process of condensing a source text into a shorter version while preserving its information content. Humans summarize on a daily basis and effortlessly, but producing high quality summaries automatically remains a challenge. The difficulty lies primarily in the nature of the task which is complex, must satisfy many constraints (e.g., summary length, informativeness, coherence, grammaticality) and ultimately requires wide-coverage text understanding. Since the latter is beyond the capabilities of current NLP technology, most work today focuses on extractive summarization, where a summary is created simply by identifying and subsequently concatenating the most important sentences in a document.

Without a great deal of linguistic analysis, it is possible to create summaries for a wide range of documents. Unfortunately, extracts are often documents of low readability and text quality and contain much redundant information. This is in marked contrast with hand-written summaries which often combine several pieces of information from the original document (Jing, 2002) and exhibit many rewrite operations such as substitutions, insertions, deletions, or reorderings.

Sentence compression is often regarded as a promising first step towards ameliorating some of the problems associated with extractive summarization. The task is commonly expressed as a word deletion problem. It involves creating a short grammatical summary of a *single* sentence, by removing elements that are considered extraneous, while retaining the most important information (Knight and Marcu, 2002). Interfacing extractive summarization with a sentence compression module could improve the conciseness of the generated summaries and render them more informative (Jing, 2000; Lin, 2003; Zajic et al., 2007).

Despite the bulk of work on sentence compression and summarization (see Clarke and Lapata 2008 and Mani 2001 for overviews) only a handful of approaches attempt to do both in a joint model (Daumé III and Marcu, 2002; Daumé III, 2006; Lin, 2003; Martins and Smith, 2009). One reason for this might be the performance of sentence compression systems which falls short of attaining grammaticality levels of human output. For example, Clarke and Lapata (2008) evaluate a range of state-of-the-art compression systems across different domains and show that machine generated compressions are consistently perceived as worse than the human gold standard. Another reason is the summarization objective itself. If our goal is to summarize news articles, then we may be better off selecting the first n sentences of the document. This "lead" baseline may err on the side of verbosity but at least will be grammatical, and it has indeed proved extremely hard to outperform by more sophisticated methods (Nenkova, 2005).

In this paper we propose a model for sum-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 565–574,
Uppsala, Sweden, 11-16 July 2010.

marization that incorporates compression into the task. A key insight in our approach is to formulate summarization as a *phrase* rather than *sentence* extraction problem. Compression falls naturally out of this formulation as only phrases deemed important should appear in the summary. Obviously, our output summaries must meet additional requirements such as sentence length, overall length, topic coverage and, importantly, grammaticality. We combine phrase and dependency information into a single data structure, which allows us to express grammaticality as constraints across phrase dependencies. We encode these constraints through the use of integer linear programming (ILP), a well-studied optimization framework that is able to search the entire solution space efficiently.

We apply our model to the task of generating highlights for a single document. Examples of CNN news articles with human-authored highlights are shown in Table 1. Highlights give a brief overview of the article to allow readers to quickly gather information on stories, and usually appear as bullet points. Importantly, they represent the gist of the *entire* document and thus often differ substantially from the first n sentences in the article (Svore et al., 2007). They are also highly compressed, written in a telegraphic style and thus provide an excellent testbed for models that generate compressed summaries. Experimental results show that our model's output is comparable to hand-written highlights both in terms of grammaticality and informativeness.

2 Related work

Much effort in automatic summarization has been devoted to sentence extraction which is often formalized as a classification task (Kupiec et al., 1995). Given appropriately annotated training data, a binary classifier learns to predict for each document sentence if it is worth extracting. Surface-level features are typically used to single out important sentences. These include the presence of certain key phrases, the position of a sentence in the original document, the sentence length, the words in the title, the presence of proper nouns, etc. (Mani, 2001; Sparck Jones, 1999).

Relatively little work has focused on extraction methods for units smaller than sentences. Jing and McKeown (2000) first extract sentences, then remove redundant phrases, and use (manual) recombination rules to produce coherent output. Wan and Paris (2008) segment sentences heuristically into clauses *before* extraction takes place, and show that this improves summarization quality. In the context of multiple-document summarization, heuristics have also been used to remove parenthetical information (Conroy et al., 2004; Siddharthan et al., 2004). Witten et al. (1999) (among others) extract keyphrases to capture the gist of the document, without however attempting to reconstruct sentences or generate summaries.

A few previous approaches have attempted to interface sentence compression with summarization. A straightforward way to achieve this is by adopting a two-stage architecture (e.g., Lin 2003) where the sentences are first extracted and then compressed or the other way round. Other work implements a *joint* model where words and sentences are deleted simultaneously from a document. Using a noisy-channel model, Daumé III and Marcu (2002) exploit the discourse structure of a document and the syntactic structure of its sentences in order to decide which constituents to drop but also which discourse units are unimportant. Martins and Smith (2009) formulate a joint sentence extraction and summarization model as an ILP. The latter optimizes an objective function consisting of two parts: an extraction component, essentially a non-greedy variant of maximal marginal relevance (McDonald, 2007), and a sentence compression component, a more compact reformulation of Clarke and Lapata (2008) based on the output of a dependency parser. Compression and extraction models are trained separately in a max-margin framework and then interpolated. In the context of multi-document summarization, Daumé III's (2006) vine-growth model creates summaries incrementally, either by starting a new sentence or by growing already existing ones.

Our own work is closest to Martins and Smith (2009). We also develop an ILP-based compression and summarization model, however, several key differences set our approach apart. Firstly, content selection is performed at the phrase rather than sentence level. Secondly, the combination of phrase and dependency information into a single data structure is new, and important in allowing us to express grammaticality as constraints across phrase dependencies, rather than resorting to a lan-

Most blacks say MLK's vision fulfilled, poll finds	**9/11 billboard draws flak from Florida Democrats, GOP**
WASHINGTON (CNN) – More than two-thirds of African-Americans believe Martin Luther King Jr.'s vision for race relations has been fulfilled, a CNN poll found – a figure up sharply from a survey in early 2008. The CNN-Opinion Research Corp. survey was released Monday, a federal holiday honoring the slain civil rights leader and a day before Barack Obama is to be sworn in as the first black U.S. president. The poll found 69 percent of blacks said King's vision has been fulfilled in the more than 45 years since his 1963 'I have a dream' speech – roughly double the 34 percent who agreed with that assessment in a similar poll taken last March. But whites remain less optimistic, the survey found.	(CNN) – A Florida man is using billboards with an image of the burning World Trade Center to encourage votes for a Republican presidential candidate, drawing criticism for politicizing the 9/11 attacks. 'Please Don't Vote for a Democrat' reads the type over the picture of the twin towers after hijacked airliners hit them on September, 11, 2001. Mike Meehan, a St. Cloud, Florida, businessman who paid to post the billboards in the Orlando area, said former President Clinton should have put a stop to Osama bin Laden and al Qaeda before 9/11. He said a Republican president would have done so.
• *69 percent of blacks polled say Martin Luther King Jr's vision realized.* • *Slim majority of whites say King's vision not fulfilled.* • *King gave his "I have a dream" speech in 1963.*	• *Billboards use image from 9/11 to encourage GOP votes.* • *9/11 image wrong for ad, say Florida political parties.* • *Floridian praises President Bush, says ex-President Clinton failed to stop al Qaeda.*

Table 1: Two example CNN news articles, showing the title and the first few paragraphs, and below, the original highlights that accompanied each story.

guage model. Lastly, our model is more compact, has fewer parameters, and does not require two training procedures. Our approach bears some resemblance to headline generation (Dorr et al., 2003; Banko et al., 2000), although we output several sentences rather than a single one. Headline generation models typically extract individual words from a document to produce a very short summary, whereas we extract phrases and ensure that they are combined into grammatical sentences through our ILP constraints.

Svore et al. (2007) were the first to foreground the highlight generation task which we adopt as an evaluation testbed for our model. Their approach is however a purely extractive one. Using an algorithm based on neural networks and third-party resources (e.g., news query logs and Wikipedia entries) they rank sentences and select the three highest scoring ones as story highlights. In contrast, we aim to generate rather than extract highlights. As a first step we focus on deleting extraneous material, but other more sophisticated rewrite operations (e.g., Cohn and Lapata 2009) could be incorporated into our framework.

3 The Task

Given a document, we aim to produce three or four short sentences covering its main topics, much like the "Story Highlights" accompanying the (online) CNN news articles. CNN highlights are written by humans; we aim to do this automatically.

	Documents	Highlights
Sentences	37.2 ± 39.6	3.5 ± 0.5
Tokens	795.0 ± 744.8	47.0 ± 9.6
Tokens/sentence	22.4 ± 4.2	13.3 ± 1.7

Table 2: Overview statistics on the corpus of documents and highlights (mean and standard deviation). A minority of documents are transcripts of interviews and speeches, and can be very long; this accounts for the very large standard deviation.

Two examples of a news story and its associated highlights, are shown in Table 1. As can be seen, the highlights are written in a compressed, almost telegraphic manner. Articles, auxiliaries and forms of the verb *be* are often deleted. Compression is also achieved through paraphrasing, e.g., substitutions and reorderings. For example, the document sentence "*The poll found 69 percent of blacks said King's vision has been fulfilled.*" is rephrased in the highlight as "*69 percent of blacks polled say Martin Luther King Jr's vision realized.*". In general, there is a fair amount of lexical overlap between document sentences and highlights (42.44%) but the correspondence between document sentences and highlights is not always one-to-one. In the first example in Table 1, the second paragraph gives rise to two highlights. Also note that the highlights need not form a coherent summary, each of them is relatively stand-alone, and there is little co-referencing between them.

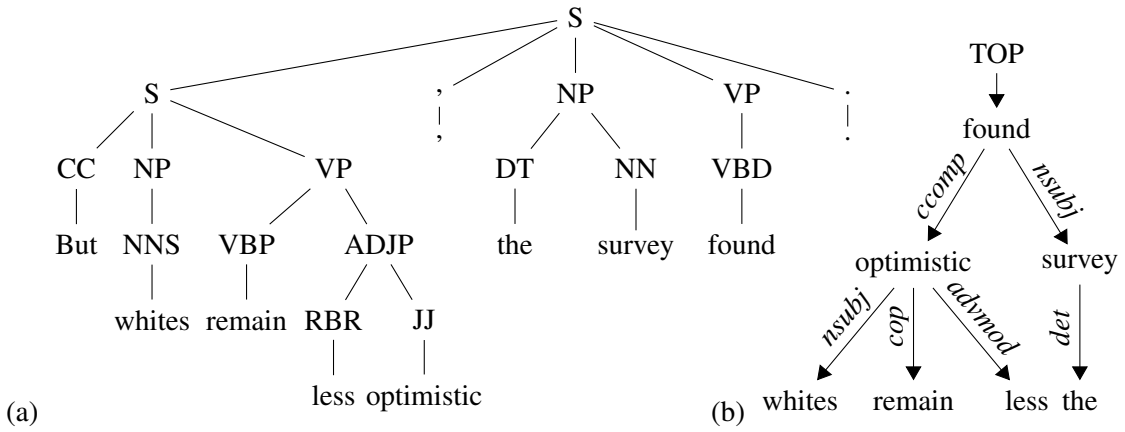


Figure 1: An example phrase structure (a) and dependency (b) tree for the sentence "*But whites remain less optimistic, the survey found.*".

In order to train and evaluate the model presented in the following sections we created a corpus of document-highlight pairs (approximately 9,000) which we downloaded from the CNN.com website.[1] The articles were randomly sampled from the years 2007–2009 and covered a wide range of topics such as business, crime, health, politics, showbiz, etc. The majority were news articles, but the set also contained a mixture of editorials, commentary, interviews and reviews. Some overview statistics of the corpus are shown in Table 2. Overall, we observe a high degree of compression both at the document and sentence level. The highlights summary tends to be ten times shorter than the corresponding article. Furthermore, individual highlights have almost half the length of document sentences.

4 Modeling

The objective of our model is to create the most informative story highlights possible, subject to constraints relating to sentence length, overall summary length, topic coverage, and grammaticality. These constraints are *global* in their scope, and cannot be adequately satisfied by optimizing each one of them individually. Our approach therefore uses an ILP formulation which will provide a globally optimal solution, and which can be efficiently solved using standard optimization tools. Specifically, the model selects phrases from which to form the highlights, and each highlight is created from a single sentence through phrase deletion. The model operates on parse trees augmented with dependency labels. We first describe how we obtain this representation and then move on to discuss the model in more detail.

Sentence Representation We obtain syntactic information by parsing every sentence twice, once with a phrase structure parser and once with a dependency parser. The phrase structure and dependency-based representations for the sentence "*But whites remain less optimistic, the survey found.*" (from Table 1) are shown in Figures 1(a) and 1(b), respectively.

We then combine the output from the two parsers, by mapping the dependencies to the edges of the phrase structure tree in a greedy fashion, shown in Figure 2(a). Starting at the top node of the dependency graph, we choose a node i and a dependency arc to node j. We locate the corresponding words i and j on the phrase structure tree, and locate their nearest shared ancestor p. We assign the label of the dependency $i \rightarrow j$ to the first unlabeled edge from p to j in the phrase structure tree. Edges assigned with dependency labels are shown as dashed lines. These edges are important to our formulation, as they will be represented by binary decision variables in the ILP. Further edges from p to j, and all the edges from p to i, are marked as *fixed* and shown as solid lines. In this way we keep the correct ordering of leaf nodes. Finally, leaf nodes are merged into parent phrases, until each phrase node contains a minimum of two tokens, shown in Figure 2(b). Because of this minimum length rule, it is possible for a merged node to be a clause rather than a phrase, but in the subsequent description we will use the term *phrase* rather loosely to describe any merged leaf node.

[1] The corpus is available from `http://homepages.inf.ed.ac.uk/mlap/resources/index.html`.

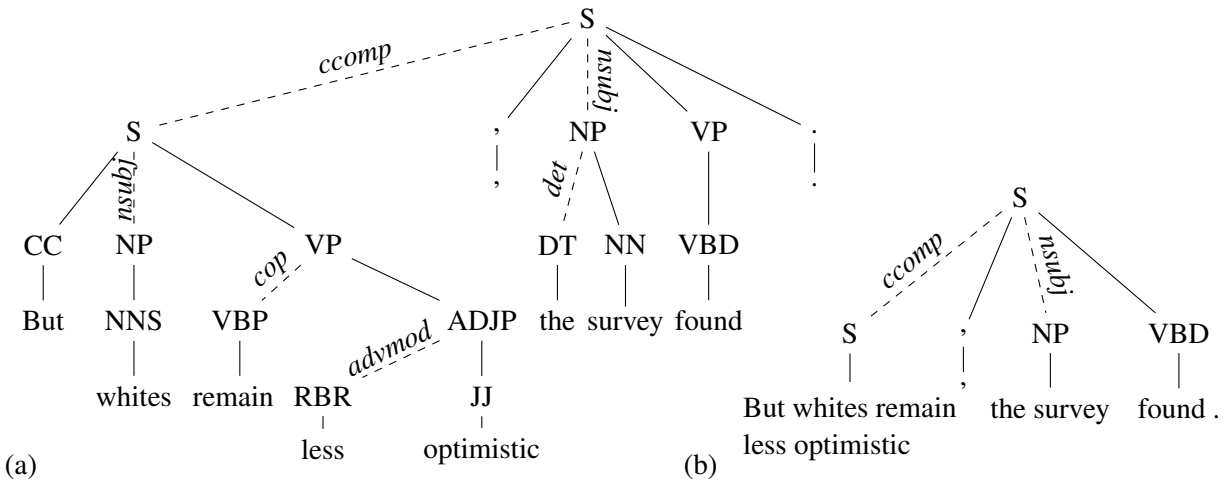


Figure 2: Dependencies are mapped onto phrase structure tree (a) and leaf nodes are merged with parent phrases (b).

ILP model The merged phrase structure tree, such as shown in Figure 2(b), is the actual input to our model. Each phrase in the document is given a *salience score*. We obtain these scores from the output of a supervised machine learning algorithm that predicts for each phrase whether it should be included in the highlights or not (see Section 5 for details). Let $\mathcal{S}$ be the set of sentences in a document, $\mathcal{P}$ be the set of phrases, and $\mathcal{P}_s \subset \mathcal{P}$ be the set of phrases in each sentence $s \in \mathcal{S}$. $\mathcal{T}$ is the set of words with the highest tf.idf scores, and $\mathcal{P}_t \subset \mathcal{P}$ is the set of phrases containing the token $t \in \mathcal{T}$. Let f_i denote the salience score for phrase i, determined by the machine learning algorithm, and l_i is its length in tokens.

We use a vector of binary variables $x \in \{0,1\}^{|\mathcal{P}|}$ to indicate if each phrase is to be within a highlight. These are either top-level nodes in our merged tree representation, or nodes whose edge to the parent has a dependency label (the dashed lines). Referring to our example in Figure 2(b), binary variables would be allocated to the top-level S node, the child S node and the NP node. The vector of auxiliary binary variables $y \in \{0,1\}^{|\mathcal{S}|}$ indicates from which sentences the chosen phrases come (see Equations (1i) and (1j)). Let the sets $\mathcal{D}_i \subset \mathcal{P}, \forall i \in \mathcal{P}$ capture the phrase dependency information for each phrase i, where each set $\mathcal{D}_i$ contains the phrases that depend on the presence of i. Our objective function function is given in Equation (1a): it is the sum of the salience scores of all the phrases chosen to form the highlights of a given document, subject to the constraints in Equations (1b)–(1j). The latter provide a natural way of describing the requirements the output must meet.

$$\max_x \quad \sum_{i \in \mathcal{P}} f_i x_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{P}} l_i x_i \leq L_T \tag{1b}$$

$$\sum_{i \in \mathcal{P}_s} l_i x_i \leq L_M y_s \qquad \forall s \in \mathcal{S} \tag{1c}$$

$$\sum_{i \in \mathcal{P}_s} l_i x_i \geq L_m y_s \qquad \forall s \in \mathcal{S} \tag{1d}$$

$$\sum_{i \in \mathcal{P}_t} x_i \geq 1 \qquad \forall t \in \mathcal{T} \tag{1e}$$

$$x_j \rightarrow x_i \qquad \forall i \in \mathcal{P}, j \in \mathcal{D}_i \tag{1f}$$

$$x_i \rightarrow y_s \qquad \forall s \in \mathcal{S}, i \in \mathcal{P}_s \tag{1g}$$

$$\sum_{s \in \mathcal{S}} y_s \leq N_S \tag{1h}$$

$$x_i \in \{0,1\} \qquad \forall i \in \mathcal{P} \tag{1i}$$

$$y_s \in \{0,1\} \qquad \forall s \in \mathcal{S}. \tag{1j}$$

Constraint (1b) ensures that the generated highlights do not exceed a total budget of L_T tokens. This constraint may vary depending on the application or task at hand. Highlights on a small screen device would presumably be shorter than highlights for news articles on the web. It is also possible to set the length of each highlight to be within the range $[L_m, L_M]$. Constraints (1c) and (1d) enforce this requirement. In particular, these constraints stop highlights formed from sentences at the beginning of the document (which tend to have

high salience scores) from being too long. Equation (1e) is a set-covering constraint, requiring that each of the words in $\mathcal{T}$ appears at least once in the highlights. We assume that words with high tf.idf scores reveal to a certain extent what the document is about. Constraint (1e) ensures that some of these words will be present in the highlights.

We enforce grammatical correctness through constraint (1f) which ensures that the phrase dependencies are respected. Phrases that depend on phrase i are contained in the set $\mathcal{D}_i$. Variable x_i is true, and therefore phrase i will be included, if any of its dependents $x_j \in \mathcal{D}_i$ are true. The phrase dependency constraints, contained in the set $\mathcal{D}_i$ and enforced by (1f), are the result of two rules based on the typed dependency information:

1. Any child node j of the current node i, whose connecting edge $i \rightarrow j$ is of type *nsubj* (nominal subject), *nsubjpass* (passive nominal subject), *dobj* (direct object), *pobj* (preposition object), *infmod* (infinitival modifier), *ccomp* (clausal complement), *xcomp* (open clausal complement), *measure* (measure phrase modifier) and *num* (numeric modifier) must be included if node i is included.

2. The parent node p of the current node i must always be included if i is, unless the edge $p \rightarrow i$ is of type *ccomp* (clausal complement) or *advcl* (adverbial clause), in which case it is possible to include i without including p.

Consider again the example in Figure 2(b). There are only two possible outputs from this sentence. If the phrase "*the survey*" is chosen, then the parent node "*found*" will be included, and from our first rule the *ccomp* phrase must also be included, which results in the output: "*But whites remain less optimistic, the survey found.*" If, on the other hand, the clause "*But whites remain less optimistic*" is chosen, then due to our second rule there is no constraint that forces the parent phrase "*found*" to be included in the highlights. Without other factors influencing the decision, this would give the output: "*But whites remain less optimistic.*" We can see from this example that encoding the possible outputs as decisions on branches of the phrase structure tree provides a more compact representation of many options than would be possible with an explicit enumeration of all possible compressions. Which output is chosen (if any) depends on the scores of the phrases involved, and the influence of the other constraints.

Constraint (1g) tells the ILP to create a highlight if one of its constituent phrases is chosen. Finally, note that a maximum number of highlights N_S can be set beforehand, and (1h) limits the highlights to this maximum.

5 Experimental Set-up

Training We obtained phrase-based salience scores using a supervised machine learning algorithm. 210 document-highlight pairs were chosen randomly from our corpus (see Section 3). Two annotators manually aligned the highlights and document sentences. Specifically, each sentence in the document was assigned one of three alignment labels: must be in the summary (1), could be in the summary (2), and is not in the summary (3). The annotators were asked to label document sentences whose content was identical to the highlights as "must be in the summary", sentences with partially overlapping content as "could be in the summary" and the remainder as "should not be in the summary". Inter-annotator agreement was .82 ($p < 0.01$, using Spearman's ρ rank correlation). The mapping of sentence labels to phrase labels was unsupervised: if the phrase came from a sentence labeled (1), and there was a unigram overlap (excluding stop words) between the phrase and any of the original highlights, we marked this phrase with a positive label. All other phrases were marked negative.

Our feature set comprised surface features such as sentence and paragraph position information, POS tags, unigram and bigram overlap with the title, and whether high-scoring tf.idf words were present in the phrase (66 features in total). The 210 documents produced a training set of 42,684 phrases (3,334 positive and 39,350 negative). We learned the feature weights with a linear SVM, using the software SVM-OOPS (Woodsend and Gondzio, 2009). This tool gave us directly the feature weights as well as support vector values, and it allowed different penalties to be applied to positive and negative misclassifications, enabling us to compensate for the unbalanced data set. The penalty hyper-parameters chosen were the ones that gave the best F-scores, using 10-fold validation.

Highlight generation We generated highlights for a test set of 600 documents. We created and

solved an ILP for each document. Sentences were first tokenized to separate words and punctuation, then parsed to obtain phrases and dependencies as described in Section 4 using the Stanford parser (Klein and Manning, 2003). For each phrase, features were extracted and salience scores calculated from the feature weights determined through SVM training. The distance from the SVM hyperplane represents the salience score. The ILP model (see Equation (1)) was parametrized as follows: the maximum number of highlights N_S was 4, the overall limit on length L_T was 75 tokens, the length of each highlight was in the range of $[8,28]$ tokens, and the topic coverage set $\mathcal{T}$ contained the top 5 tf.idf words. These parameters were chosen to capture the properties seen in the majority of the training set; they were also relaxed enough to allow a feasible solution of the ILP model (with hard constraints) for all the documents in the test set. To solve the ILP model we used the ZIB Optimization Suite software (Achterberg, 2007; Koch, 2004; Wunderling, 1996). The solution was converted into highlights by concatenating the chosen leaf nodes in order. The ILP problems we created had on average 290 binary variables and 380 constraints. The mean solve time was 0.03 seconds.

Summarization In order to examine the generality of our model and compare with previous work, we also evaluated our system on a vanilla summarization task. Specifically, we used the same model (trained on the CNN corpus) to generate summaries for the DUC-2002 corpus[2]. We report results on the entire dataset and on a subset containing 140 documents. This is the same partition used by Martins and Smith (2009) to evaluate their ILP model.[3]

Baselines We compared the output of our model to two baselines. The first one simply selects the "leading" three sentences from each document (without any compression). The second baseline is the output of a sentence-based ILP model, similar to our own, but simpler. The model is given in (2). The binary decision variables $x \in \{0,1\}^{|\mathcal{S}|}$ now represent sentences, and f_i the salience score for each sentence. The objective again is to maximize the total score, but now subject only to tf.idf coverage (2b) and a limit on the number of highlights (2c) which we set to 3. There are no sentence length or grammaticality constraints, as there is no sentence compression.

$$\max_{x} \quad \sum_{i \in \mathcal{S}} f_i x_i \tag{2a}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{S}_t} x_i \geq 1 \qquad \forall t \in \mathcal{T} \tag{2b}$$

$$\sum_{i \in \mathcal{S}} x_i \leq N_S \tag{2c}$$

$$x_i \in \{0,1\} \qquad \forall i \in \mathcal{S}. \tag{2d}$$

The SVM was trained with the same features used to obtain phrase-based salience scores, but with sentence-level labels (labels (1) and (2) positive, (3) negative).

Evaluation We evaluated summarization quality using ROUGE (Lin and Hovy, 2003). For the highlight generation task, the original CNN highlights were used as the reference. We report unigram overlap (ROUGE-1) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

In addition, we evaluated the generated highlights by eliciting human judgments. Participants were presented with a news article and its corresponding highlights and were asked to rate the latter along three dimensions: informativeness (do the highlights represent the article's main topics?), grammaticality (are they fluent?), and verbosity (are they overly wordy and repetitive?). The subjects used a seven point rating scale. An ideal system would receive high numbers for grammaticality and informativeness and a low number for verbosity. We randomly selected nine documents from the test set and generated highlights with our model and the sentence-based ILP baseline. We also included the original highlights as a gold standard. We thus obtained ratings for 27 (9×3) document-highlights pairs.[4] The study was conducted over the Internet using WebExp (Keller et al., 2009) and was completed by 34 volunteers, all self reported native English speakers.

With regard to the summarization task, following Martins and Smith (2009), we used ROUGE-1 and ROUGE-2 to evaluate our system's output. We also report results with ROUGE-L. Each document in the DUC-2002 dataset is paired with

[2] http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

[3] We are grateful to André Martins for providing us with details of their testing partition.

[4] A Latin square design ensured that subjects did not see two different highlights of the same document.

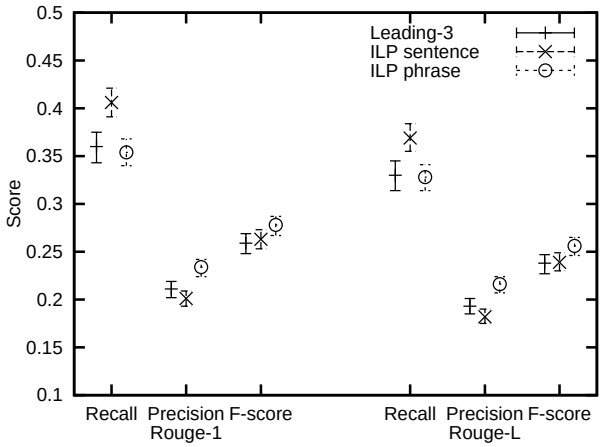


Figure 3: ROUGE-1 and ROUGE-L results for phrase-based ILP model and two baselines, with error bars showing 95% confidence levels.

a human-authored summary (approximately 100 words) which we used as reference.

6 Results

We report results on the highlight generation task in Figure 3 with ROUGE-1 and ROUGE-L (error bars indicate the 95% confidence interval). In both measures, the ILP sentence baseline has the best recall, while the ILP phrase model has the best precision (the differences are statistically significant). F-score is higher for the phrase-based system but not significantly. This can be attributed to the fact that the longer output of the sentence-based model makes the recall task easier. Average highlight lengths are shown in Table 3, and the compression rates they represent. Our phrase model achieves the highest compression rates, whereas the sentence-based model tends to select long sentences even in comparison to the lead baseline. The sentence ILP model outperforms the lead baseline with respect to recall but not precision or F-score. The phrase ILP achieves a significantly better F-score over the lead baseline with both ROUGE-1 and ROUGE-L.

The results of our human evaluation study are summarized in Table 4. There was no statistically significant difference in the grammaticality between the highlights generated by the phrase ILP system and the original CNN highlights (means differences were compared using a Post-hoc Tukey test). The grammaticality of the sentence ILP was significantly higher overall as no compression took place ($\alpha < 0.05$). All three

	s	toks/s	C.R.
Articles	36.5	22.2 ± 4.0	100%
CNN highlights	3.5	13.3 ± 1.7	5.8%
ILP phrase	3.8	18.0 ± 2.9	8.4%
Leading-3	3.0	25.1 ± 7.4	9.3%
ILP sentence	3.0	31.3 ± 7.9	11.6%

Table 3: Comparison of output lengths: number of sentences, tokens per sentence, and compression rate, for CNN articles, their highlights, the ILP phrase model, and two baselines.

Model	Grammar	Importance	Verbosity
CNN highlights	4.85	4.88	3.14
ILP sentence	6.41	5.47	3.97
ILP phrase	5.53	5.05	3.38

Table 4: Average human ratings for original CNN highlights, and two ILP models.

systems performed on a similar level with respect to importance (differences in the means were not significant). The highlights created by the sentence ILP were considered significantly more verbose ($\alpha < 0.05$) than those created by the phrase-based system and the CNN abstractors. Overall, the highlights generated by the phrase ILP model were not significantly different from those written by humans. They capture the same content as the full sentences, albeit in a more succinct manner. Table 5 shows the output of the phrase-based system for the documents in Table 1.

Our results on the complete DUC-2002 corpus are shown in Table 6. Despite the fact that our model has not been optimized for the original task of generating 100-word summaries—instead it is trained on the CNN corpus, and generates highlights—the results are comparable with the best of the original participants[5] in each of the ROUGE measures. Our model is also significantly better than the lead sentences baseline.

Table 7 presents our results on the same DUC-2002 partition (140 documents) used by Martins and Smith (2009). The phrase ILP model achieves a significantly better F-score (for both ROUGE-1 and ROUGE-2) over the lead baseline, the sentence ILP model, and Martins and Smith. We should point out that the latter model is not a straw man. It significantly outperforms a pipeline

[5]The list of participants is on page 12 of the slides available from `http://duc.nist.gov/pubs/2002slides/overview.02.pdf`.

- *More than two-thirds of African-Americans believe Martin Luther King Jr.'s vision for race relations has been fulfilled.*
- *69 percent of blacks said King's vision has been fulfilled in the more than 45 years since his 1963 'I have a dream' speech.*
- *But whites remain less optimistic, the survey found.*

- *A Florida man is using billboards with an image of the burning World Trade Center to encourage votes for a Republican presidential candidate, drawing criticism.*
- *'Please Don't Vote for a Democrat' reads the type over the picture of the twin towers.*
- *Mike Meehan said former President Clinton should have put a stop to Osama bin Laden and al Qaeda before 9/11.*

Table 5: Generated highlights for the stories in Table 1 using the phrase ILP model.

Participant	ROUGE-1	ROUGE-2	ROUGE-L
28	0.464	0.222	0.432
19	0.459	0.221	0.431
21	0.458	0.216	0.426
29	0.449	0.208	0.419
27	0.445	0.209	0.417
Leading-3	0.416	0.200	0.390
ILP phrase	0.454	0.213	0.428

Table 6: ROUGE results on the complete DUC-2002 corpus, including the top 5 original participants. For all results, the 95% confidence interval is $\pm$0.008.

approach that first creates extracts and then compresses them. Furthermore, as a standalone sentence compression system it yields state of the art performance, comparable to McDonald's (2006) discriminative model and superior to Hedge Trimmer (Zajic et al., 2007), a less sophisticated deterministic system.

7 Conclusions

In this paper we proposed a joint content selection and compression model for single-document summarization. A key aspect of our approach is the representation of content by phrases rather than entire sentences. Salient phrases are selected to form the summary. Grammaticality, length and coverage requirements are encoded as constraints in an integer linear program. Applying the model to the generation of "story highlights" (and single document summaries) shows that it is a viable alternative to extraction-based systems. Both ROUGE scores and the results of our human study

	ROUGE-1	ROUGE-2	ROUGE-L
Leading-3	.400 $\pm$ .018	.184 $\pm$ .015	.374 $\pm$ .017
M&S (2009)	.403 $\pm$ .076	.180 $\pm$ .076	—
ILP sentence	.430 $\pm$ .014	.191 $\pm$ .015	.401 $\pm$ .014
ILP phrase	.445 $\pm$ .014	.200 $\pm$ .014	.419 $\pm$ .014

Table 7: ROUGE results on DUC-2002 corpus (140 documents). —: only ROUGE-1 and ROUGE-2 results are given in Martins and Smith (2009).

confirm that our system manages to create summaries at a high compression rate and yet maintain the informativeness and grammaticality of a competitive extractive system. The model itself is relatively simple and knowledge-lean, and achieves good performance without reference to any resources outside the corpus collection.

Future extensions are many and varied. An obvious next step is to examine how the model generalizes to other domains and text genres. Although coherence is not so much of an issue for highlights, it certainly plays a role when generating standard summaries. The ILP model can be straightforwardly augmented with discourse constraints similar to those proposed in Clarke and Lapata (2007). We would also like to generalize the model to arbitrary rewrite operations, as our results indicate that compression rates are likely to improve with more sophisticated paraphrasing.

Acknowledgments

We would like to thank Andreas Grothey and members of ICCS at the School of Informatics for the valuable discussions and comments throughout this work. We acknowledge the support of EPSRC through project grants EP/F055765/1 and GR/T04540/01.

References

Achterberg, Tobias. 2007. *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin.

Banko, Michele, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th ACL*. Hong Kong, pages 318–325.

Clarke, James and Mirella Lapata. 2007. Modelling compression with discourse constraints. In *Proceedings of EMNLP-CoNLL*. Prague, Czech Republic, pages 1–11.

Clarke, James and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* 31:399–429.

Cohn, Trevor and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research* 34:637–674.

Conroy, J. M., J. D. Schlesinger, J. Goldstein, and D. P. O'Leary. 2004. Left-brain/right-brain multi-document summarization. In *DUC 2004 Conference Proceedings*.

Daumé III, Hal. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Daumé III, Hal and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th ACL*. Philadelphia, PA, pages 449–456.

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization*. pages 1–8.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th ANLP*. Seattle, WA, pages 310–315.

Jing, Hongyan. 2002. Using hidden Markov modeling to decompose human-written summaries. *Computational Linguistics* 28(4):527–544.

Jing, Hongyan and Kathleen McKeown. 2000. Cut and paste summarization. In *Proceedings of the 1st NAACL*. Seattle, WA, pages 178–185.

Keller, Frank, Subahshini Gunasekharan, Neil Mayo, and Martin Corley. 2009. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods* 41(1):1–12.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*. Sapporo, Japan, pages 423–430.

Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

Koch, Thorsten. 2004. *Rapid Mathematical Prototyping*. Ph.D. thesis, Technische Universität Berlin.

Kupiec, Julian, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of SIGIR-95*. Seattle, WA, pages 68–73.

Lin, Chin-Yew. 2003. Improving summarization performance by sentence compression — a pilot study. In *Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages*. Sapporo, Japan, pages 1–8.

Lin, Chin-Yew and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT NAACL*. Edmonton, Canada, pages 71–78.

Mani, Inderjeet. 2001. *Automatic Summarization*. John Benjamins Pub Co.

Martins, André and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*. Boulder, Colorado, pages 1–9.

McDonald, Ryan. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of the 11th EACL*. Trento, Italy.

McDonald, Ryan. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th ECIR*. Rome, Italy.

Nenkova, Ani. 2005. Automatic text summarization of newswire: Lessons learned from the Document Understanding Conference. In *Proceedings of the 20th AAAI*. Pittsburgh, PA, pages 1436–1441.

Siddharthan, Advaith, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. pages 896–902.

Sparck Jones, Karen. 1999. Automatic summarizing: Factors and directions. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, MIT Press, Cambridge, pages 1–33.

Svore, Krysta, Lucy Vanderwende, and Christopher Burges. 2007. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of EMNLP-CoNLL*. Prague, Czech Republic, pages 448–457.

Wan, Stephen and Cécile Paris. 2008. Experimenting with clause segmentation for text summarization. In *Proceedings of the 1st TAC*. Gaithersburg, MD.

Witten, Ian H., Gordon Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM International Conference on Digital Libraries*. Berkeley, CA, pages 254–255.

Woodsend, Kristian and Jacek Gondzio. 2009. Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications* .

Wunderling, Roland. 1996. *Paralleler und objektorientierter Simplex-Algorithmus*. Ph.D. thesis, Technische Universität Berlin.

Zajic, David, Bonnie J. Door, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing Management Special Issue on Summarization* 43(6):1549–1570.

Sentence and Expression Level Annotation of Opinions in User-Generated Discourse

Cigdem Toprak and Niklas Jakob and Iryna Gurevych
Ubiquitous Knowledge Processing Lab
Computer Science Department, Technische Universität Darmstadt, Hochschulstraße 10
D-64289 Darmstadt, Germany
www.ukp.tu-darmstadt.de

Abstract

In this paper, we introduce a corpus of consumer reviews from the *rateitall* and the *eopinions* websites annotated with opinion-related information. We present a two-level annotation scheme. In the first stage, the reviews are analyzed at the sentence level for *(i)* relevancy to a given topic, and *(ii)* expressing an evaluation about the topic. In the second stage, on-topic sentences containing evaluations about the topic are further investigated at the expression level for pinpointing the properties (semantic orientation, intensity), and the functional components of the evaluations (opinion terms, targets and holders). We discuss the annotation scheme, the inter-annotator agreement for different subtasks and our observations.

1 Introduction

There has been a huge interest in the automatic identification and extraction of opinions from free text in recent years. Opinion mining spans a variety of subtasks including: creating opinion word lexicons (Esuli and Sebastiani, 2006; Ding et al., 2008), identifying opinion expressions (Riloff and Wiebe, 2003; Fahrni and Klenner, 2008), identifying polarities of opinions in context (Breck et al., 2007; Wilson et al., 2005), extracting opinion targets (Hu and Liu, 2004; Zhuang et al., 2006; Cheng and Xu, 2008) and opinion holders (Kim and Hovy, 2006; Choi et al., 2005).

Data-driven approaches for extracting opinion expressions, their holders and targets require reliably annotated data at the expression level. In previous research, expression level annotation of opinions was extensively investigated on newspaper articles (Wiebe et al., 2005; Wilson and Wiebe, 2005; Wilson, 2008b) and on meeting dialogs (Somasundaran et al., 2008; Wilson, 2008a).

Compared to the newspaper and meeting dialog genres, little corpus-based work has been carried out for interpreting the opinions and evaluations in user-generated discourse. Due to the high popularity of Web 2.0 communities[1], the amount of user-generated discourse and the interest in the analysis of such discourse has increased over the last years. To the best of our knowledge, there are two corpora of user-generated discourse which are annotated for opinion related information at the expression level: The corpus of Hu & Liu (2004) consists of customer reviews about consumer electronics, and the corpus of Zhuang et al. (2006) consists of movie reviews. Both corpora are tailored for application specific needs, therefore, do not contain certain related information explicitly annotated in the discourse, which we consider important (see Section 2). Furthermore, none of these works provide inter-annotator agreement studies.

Our goal is to create sentence and expression level annotated corpus of customer reviews which fulfills the following requirements: (1) It filters individual sentences regarding their topic relevancy and the existence of an opinion or factual information which implies an evaluation. (2) It identifies opinion expressions including the respective opinion target, opinion holder, modifiers, and anaphoric expressions if applicable. (3) The semantic orientation of the opinion expression is identified while considering negation, and the opinion expression is linked to the respective holder and target in the discourse. Such a resource would *(i)* enable novel applications of opinion mining such as a fine-grained identification of opinion properties, e.g. opinion modification detection including negation, and *(ii)* enhance opinion target extraction and the polarity assignment by linking the opinion expression with its target

[1] http://blog.nielsen.com/nielsenwire/wp-content/uploads/2008/10/press_release24.pdf

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 575–584,
Uppsala, Sweden, 11-16 July 2010.

and providing anaphoric resolutions in discourse.

We present an annotation scheme which fulfills the mentioned requirements, an inter-annotator agreement study, and discuss our observations. The rest of this paper is structured as follows: Section 2 presents the related work. In Sections 3, we describe the annotation scheme. Section 4 presents the data and the annotation study, while Section 5 summarizes the main conclusions.

2 Previous Opinion Annotated Corpora

2.1 Newspaper Articles and Meeting Dialogs

Most prominent work concerning the expression level annotation of opinions is the Multi-Perspective Question Answering (MPQA) corpus[2] (Wiebe et al., 2005). It was extended several times over the last years, either by adding new documents or annotating new types of opinion related information (Wilson and Wiebe, 2005; Stoyanov and Cardie, 2008; Wilson, 2008b). The MPQA annotation scheme builds upon the *private state* notion (Quirk et al., 1985) which describes mental states including opinions, emotions, speculations and beliefs among others. The annotation scheme strives to represent the *private states* in terms of their functional components (i.e. *experiencer* holding an *attitude* towards a *target*). It consists of frames (*direct subjective, expressive subjective element, objective speech event, agent, attitude,* and *target frames*) with slots representing various attributes and properties (e.g.*intensity, nested source*) of the private states.

Wilson (2008a) adapts and extends the concepts from the MPQA scheme to annotate subjective content in meetings (AMI corpus), and creates the AMIDA scheme. Besides subjective utterances, the AMIDA scheme contains *objective polar utterances* which annotates evaluations without expressing explicit opinion expressions.

Somasundaran et al. (2008) proposes *opinion frames* for representing discourse level associations in meeting dialogs. The annotation scheme focuses on two types of opinions, *sentiment* and *arguing*. It annotates the opinion expression and target spans. The *link* and *link type* attributes associate the target with other targets in the discourse through *same* or *alternative* relations. The *opinion frames* are built based on the links between targets. Somasundaran et al. (2008) show that *opinion frames* enable a coherent interpretation of the opinions in discourse and discover implicit evaluations through link transitivity.

Similar to Somasundaran et al. (2008), Asher et al. (2008) performs discourse level analysis of opinions. They propose a scheme which first identifies and assigns categories to the opinion segments as *reporting, judgment, advice,* or *sentiment*; and then links the opinion segments with each other via rhetorical relations including *contrast, correction, support, result*, or *continuation*. However, in contrast to our scheme and other schemes, instead of marking expression boundaries without any restriction they annotate an opinion segment only if it contains an opinion word from their lexicon, or if it has a rhetorical relation to another opinion segment.

2.2 User-generated Discourse

The two annotated corpora of user-generated content and their corresponding annotation schemes are far less complex. Hu & Liu (2004) present a dataset of customer reviews for consumer electronics crawled from amazon.com. The following example shows two annotations taken from the corpus of Hu & Liu (2004):

camera[+2]##This is my first digital camera and what a toy it is...

size[+2][u]##it is small enough to fit easily in a coat pocket or purse.

The corpus provides only target and polarity annotations, and do not contain opinion expression or opinion modifier annotations which lead to these polarity scores. The annotation scheme allows the annotation of implicit features (indicated with the the attribute *[u]*). Implicit features are not resolved to any actual product feature instances in discourse. In fact, the actual positions of the product features (or any anaphoric references to them) are not explicitly marked in the discourse, i.e, it is unclear to which mention of the feature the opinion refers to.

In their paper on movie review mining and summarization, Zhuang et al. (2006) introduce an annotated corpus of movie reviews from the Internet Movie Database. The corpus is annotated regarding movie features and corresponding opinions. The following example shows an annotated sentence:

⟨Sentence⟩I have never encountered a movie whose supporting cast was so perfectly realized.⟨FO Fword="supporting cast" Ftype="PAC" Oword="perfect" Otype="PRO"/⟩⟨/Sentence⟩

[2] http://www.cs.pitt.edu/mpqa/

The movie features (*Fword*) are attributed to one of 20 predefined categories (Ftype). The opinion words (*Oword*) and their semantic orientations (*Otype*) are identified. Possible negations are directly reflected by the semantic orientation, but not explicitly labeled in the sentence. *(PD)* in the following example indicates that the movie feature is referenced by anaphora:

⟨Sentence⟩It is utter nonsense and insulting to my intelligence and sense of history. ⟨FO Fword="film(PD)" Ftype="OA" Oword="nonsense, insulting" Otype="CON"/⟩⟨/Sentence⟩

However, similar to the corpus of Hu & Liu (2004) the referring pronouns are not explicitly marked in discourse. It is therefore neither possible to automatically determine which pronoun creates the link if there are more than one in a sentence, nor it is denoted which antecedent, i.e. the actual mention of the feature in the discourse it relates to.

3 Annotation Scheme

3.1 Opinion versus Polar Facts

The goal of the annotation scheme is to capture the evaluations regarding the topics being discussed in the consumer reviews. The evaluations in consumer reviews are either explicit expressions of opinions, or facts which imply evaluations as discussed below.

Explicit expressions of opinions: Opinions are private states (Wiebe et al., 2005; Quirk et al., 1985) which are not open to objective observation or verification. In this study, we focus on the opinions stating the quality or value of an entity, experience or a proposition from one's perspective. (1) illustrates an example of an explicit expression of an opinion. Similar to Wiebe et al. (2005), we view opinions in terms of their functional components, as *opinion holders*, e.g., the author in (1), holding attitudes (*polarity*), e.g., negative attitude indicated with the word *nightmare*, towards possible *targets*, e.g., *Capella University*.

(1) I had a nightmare with Capella University.[3]

Facts implying evaluations: Besides opinions, there are facts which can be objectively verified, but still imply an evaluation of the quality or value of an entity or a proposition. For instance, consider the snippet below:

[3]We use authentic examples from the corpus without correcting grammatical or spelling errors.

(2) In a 6-week class, I counted 3 comments from the professors directly to me and two directed to my team.
(3) I found that I spent most of my time learning from my fellow students.
(4) A standard response from my professors would be that of a sentence fragment.

The example above provides an evaluation about the professors without stating any explicit expressions of opinions. We call such objectively verifiable, but evaluative sentences *polar facts*. Explicit expressions of opinions typically contain specific cues, i.e. opinion words, loaded with a positive or negative connotation (e.g., *nightmare*). Even when they are taken out of the context in which they appear, they evoke an evaluation. However, evaluations in *polar facts* can only be inferred within the context of the review. For instance, the targets of the implied evalution in the polar facts (2), (3) and (4) are the professors. However, (3) may have been perceived as a positive statement if the review was explaining how good the fellow students were or how the course enforced team work etc.

The annotation scheme consists of two levels. First, the sentence level scheme analyses each sentence in terms of *(i)* its relevancy to the overall topic of the review, and *(ii)* whether it contains an evaluation (an opinion or a *polar fact*) about the topic. Once the on-topic sentences containing evaluations are identified, the expression level scheme first focuses either on marking the text spans of the opinion expressions (if the sentence contains an explicit expression of an opinion) or marking the targets of the *polar facts* (if the sentence is a *polar fact*). Upon marking an opinion expression span, the target and holder of the opinion is marked and linked to the marked opinion expression. Furthermore, the expression level scheme allows assigning polarities to the marked opinion expression spans and targets of the *polar facts*.

The following subsections introduce the sentence and the expression level annotation schemes in detail with examples.

3.2 Sentence Level Annotation

The sentence annotation strives to identify the sentences containing evaluations about the topic. In consumer reviews people occasionally drift off the actual topic being reviewed. For instance, as in (5) taken from a review about an online university, they tend to provide information about their background or other experiences.

(5) I am very fortunate and almost right out of high school

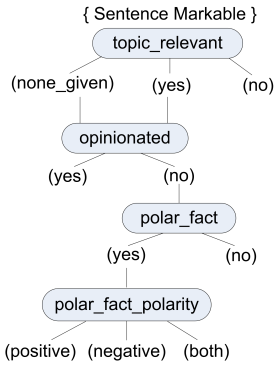


Figure 1: The sentence level annotation scheme

with a very average GPA and only 20; I already make above $45,000 a year as a programmer with a large health care company for over a year and have had 3 promotions up in the first year and a half.

Such sentences do not provide information about the actual topic, but typically serve for justifying the user's point of view or provide a better understanding about her circumstances. However, they are not valuable for an application aiming to extract opinions about a specific topic.

Reviews given to the annotators contain meta information stating the topic, for instance, the name of the university or the service being reviewed. A markable (i.e. an annotation unit) is created for each sentence prior to the annotation process. At this level, the annotation process is therefore a sentence labeling task. The annotators are able to see the whole review, and instructed to label sentences in the context of the whole review. Figure 1 presents the sentence level scheme. Attribute names are marked with oval circles and the possible values are given in parenthesis. The following attributes are used:

topic_relevant attribute is labeled as *yes* if the sentence discusses the given topic itself or its aspects, properties or features as in examples (1)-(4). Other possible values for this attribute include *none_given* which can be chosen in the absence of meta data, or *no* if the sentence drifted off the topic as in example (5).

opinionated attribute is labeled as *yes* if the sentence contains any explicit expressions of opinions about the given topic. This attribute is presented if the *topic_relevant* attribute has been labeled as *none_given* or *yes*. In other words, only the on-topic sentences are considered in this step. Examples (6)-(8) illustrate examples labeled as *topic_relevant=yes* and *opinionated=yes*.

*(6) Many people are knocking Devry but I have seen them to be a very great school. [**Topic:** Devry University]*
*(7) University of Phoenix was a surprising disappointment. [**Topic:** University of Phoenix]*
*(8) Assignments were passed down, but when asked to clarify the assignment because the syllabus had contradicting, poorly worded, information, my professors regularly responded...."refer to the syllabus"....but wait, the syllabus IS the question. [**Topic:** University of Phoenix]*

polar_fact attribute is labeled as *yes* if the sentence is a *polar fact*. This attribute is presented if the *opinionated* attribute has been labeled as *no*. Examples (2)-(4) demonstrate sentences labeled as *topic_relevant=yes, opinionated=no* and *polar_fact=yes*.

polar_fact_polarity attribute represents the polarity of the evaluation in a *polar_fact* sentence. The possible values for this attribute include *positive, negative, both*. The value *both* is intended for the *polar_fact* sentences containing more than one evaluation with contradicting polarities. At the expression level analysis, the targets of the contradicting *polar_fact* evaluations are identified distinctly and assigned polarities of *positive* or *negative* later on. Examples (9)-(11) demonstrate examples of *polar fact* sentences with different values of the attribute *polar_fact_polarity*.

*(9) There are students in the first programming class and after taking this class twice they cannot write a single line of code. **[polar_fact_polarity=negative]***
*(10) The same class (i.e. computer class) being teach at Ivy League schools are being offered at Devry. **[polar_fact_polarity=positive]***
*(11) The lectures are interactive and recorded, but you need a consent from the instructor each time. **[polar_fact_polarity=both]***

3.3 Expression Level Annotation

At the expression level, we focus on the topic relevant sentences containing evaluations, i.e., sentences labeled as *topic_relevant=yes, opinionated=yes* or *topic_relevant=yes, opinionated=no, polar_fact=yes*. If the sentence is a *polar fact*, then the aim is to mark the target and label the polarity of the evaluation. If the sentence is opinionated, then, the aim is to mark the opinion expression span, and label its polarity and strength (i.e. intensity), and to link it to the target and the holder.

Figure 2 presents the expression level scheme. At this stage, annotators mark text spans, and are allowed to assign one of the five labels to the marked span:

The ***polar_target*** is used to label the targets of the evaluations implied by *polar facts*. The *isReference* attribute labels *polar_targets* which are anaphoric references. The *polar_target_polarity*

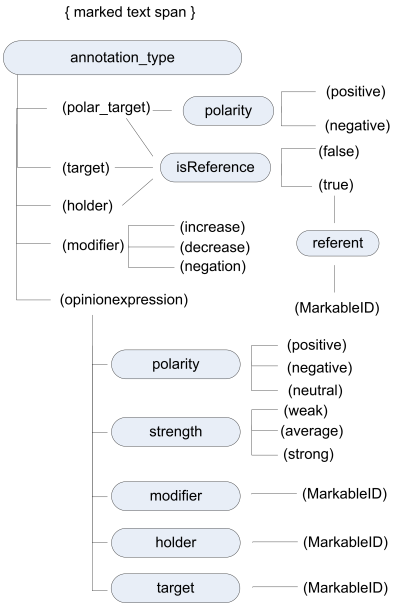


Figure 2: The expression level annotation scheme

attribute is used to label the polarity as *positive* or *negative*. If the *isReference* attribute is labeled as *true*, then the *referent* attribute appears which enables the annotator to resolve the reference to its antecedent. Consider the example sentences (12) and (13) below. The *polar_target* in (13), written bold, is labeled as *isReference=true*, *polar_target_polarity=negative*. To resolve the reference, annotator first creates another *polar_target* markable for the antecedent, namely the bold text span in (12), then, links the antecedent to the *referent* attribute of the *polar_target* in (13).

(12) Since classes already started, ***CTU*** *told me they would extend me so that I could complete the classes and get credit once I got back.*
(13) What ***they*** *didn't tell me is in order to extend, I also had to be enrolled in the next semester.*

The ***target*** annotation represents what the opinion is about. Both *polar_targets* and *targets* can be the topic of the review or different aspects, i.e. features of the topic. Similar to the *polar_targets*, the *isReference* attribute allows the identification of the targets which are anaphoric references and the *referent* attribute links them to their antecedents in the discourse. Bold span in (14) shows an example of a *target* in an opinionated sentence.

(14) Capella U has incredible ***faculty in the Harold Abel School of Psychology****.*

The ***holder*** type represents the holder of an opinion in the discourse and is labeled in the same manner as the *targets* and *polar_targets*. In consumer reviews, *holders* are most of the time the authors of the reviews. To ease the annotation process, the holder is not labeled when this is the author.

The ***modifier*** annotation labels the lexical items, such as *not, very, hardly etc.*, which affect the strength of an opinion or shift its polarity. Upon creation of a *modifier* markable, annotators are asked to choose between *negation, increase, decrease* for identifying the influence of the *modifier* on the opinion. For instance, the marked span in (15) is labeled as *modifier=increase* as it gives the impression that the author is really offended by the negative comments about her university.

(15) I am ***quite honestly*** *appauled by some of the negative comments given for Capella University on this website.*

The ***opinionexpression*** annotation is used to label the opinion terms in the sentence. This markable type has five attributes, three of which, i.e., *modifier*, *holder*, and *target* are pointer attributes to the previously defined markable types. The *polarity* attribute assesses the semantic orientation of the attitude, where the *strength* attribute marks the intensity of this attitude. The *polarity* and *strength* attributes focus solely on the marked *opinionexpression* span, not the whole evaluation implied in the sentence. For instance, the *opinionexpression* span in (16) is labeled as *polarity=negative, strength=average*. We infer the polarity of the evaluation only after considering the *modifier, polarity* and the *strength* attributes together. In (16), the evaluation about the target is strongly negative after considering all three attributes of the *opinionexpression* annotation. In (17), the *polarity* of the *opinionexpression1* itself (*complaints*) is labeled as *negative*. It is linked to the *modifier1* which is labeled as *negation*. *Target1* (*PhD journey*) is linked to the *opinionexpression1*. The overall evaluation regarding the *target1* is positive after applying the affect of the *modifier1* to the *polarity* of the *opinionexpression1*, i.e., after negating the negative polarity.

(16) I am ***quite honestly*** $_{[modifier]}$ ***appauled by*** $_{[opinionexpression]}$ *some of the* ***negative comments given for Capella University on this website*** $_{[target]}$.
(17) I have ***no*** $_{[modifier1]}$ ***complaints*** $_{[opinionexpression1]}$ *about the entire* ***PhD journey*** $_{[target1]}$ *and* ***highly*** $_{[modifier2]}$ ***recommend*** $_{[opinionexpression2]}$ ***this school*** $_{[target2]}$.

Finally, Figure 3 demonstrates all expression level markables created for an opinionated sentence and how they relate to each other.

Figure 3: Expression level annotation example

4 Annotation Study

Each review has been annotated by two annotators independently according to the annotation scheme introduced above. We used the freely available MMAX2[4] annotation tool capable of stand-off multi-level annotations. Annotators were native speaker linguistic students. They were trained on 15 reviews after reading the annotation manual.[5] In the training stage, the annotators discussed with each other if different decisions have been made and were allowed to ask questions to clarify their understanding of the scheme. Annotators had access to the review text as a whole while making their decisions.

4.1 Data

The corpus consists of consumer reviews collected from the review portals *rateitall*[6] and *epinions*[7]. It contains reviews from two domains including online universities, e.g., Capella University, Pheonix, University of Maryland University College etc. and online services, e.g., PayPal, egroups, eTrade, eCircles etc. These two domains were selected with the project-relevant, domain-specific research goals in mind. We selected a specific topic, e.g. Pheonix, if there were more than 3 reviews written about it. Table 1 shows descriptive statistics regarding the data.

We used 118 reviews containing 1151 sentences from the university domain for measuring the sentence and expression level agreements. In the following subsections, we report the inter-annotator agreement (IAA) at each level.

[4] http://mmax2.sourceforge.net/

[5] http://www.ukp.tu-darmstadt.de/research/data/sentiment-analysis

[6] http://www.rateitall.com

[7] http://www.epinions.com

	University	Service	All
Reviews	240	234	474
Sentences	2786	6091	8877
Words	49624	102676	152300
Avg sent./rev.	11.6	26	18.7
Std. dev. sent./rev.	8.2	16	14.6
Avg. words/rev.	206.7	438.7	321.3
Std. dev. words/rev.	159.2	232.1	229.8

Table 1: Descriptive statistics about the corpus

4.2 Sentence Level Agreement

Sentence level markables were already created automatically prior to the annotation, i.e., the set of annotation units were the same for both annotators. We use Cohen's kappa (κ) (Cohen, 1960) for measuring the IAA. The sentence level annotation scheme has a hierarchical structure. A new attribute is presented based on the decision made for the previous attribute, for instance, *opinionated* attribute is only presented if the *topic_relevant* attribute is labeled as *yes* or *none_given*; *polar_fact* attribute is only presented if the *opinionated* attribute is labeled as *no* etc. We calculate κ for each attribute considering only the markables which were labeled the same by both annotators in the previously required step. Table 2 shows the κ values for each attribute, the size of the markable set on which the value was calculated, and the percentage agreement.

Attribute	Markables	Agr.	κ
topic_relevant	1151	0.89	0.73
opinionated	682	0.80	0.61
polar_fact	258	0.77	0.56
polar_fact_polarity	103	0.96	0.92

Table 2: Sentence level inter-annotator agreement

The agreement for topic relevancy shows that it is possible to label this attribute reliably. The sentences labeled as topic relevant by both annotators correspond to 59% of all sentences, suggesting that people often drift off the topic in consumer reviews. This is usually the case when they provide information about their backgrounds or alternatives to the given topic.

On the other hand, we obtain moderate agreement levels for the *opinionated* and *polar_fact* attributes. 62% of the topic relevant sentences were labeled as opinionated by at least one annotator, and the rest 38% constitute the topic relevant sentences labeled as not opinionated by both annotators. Nonetheless, they still contain evaluations (*polar_facts*), as 15% of the topic relevant sen-

tences were labeled as *polar_facts* by both annotators. When we merge the attributes *opinionated* and *polar_fact* into a single category, we obtain κ of 0.75 and a percentage agreement of 87%. Thus, we conclude that opinion-relevant sentences, either in the form of an explicit expression of opinion or a *polar fact*, can be labeled reliably in consumer reviews. However, there is a thin border between *polar facts* and explicit expressions of opinions.

To the best of our knowledge, similar annotation efforts on consumer or movie reviews do not provide any agreement figures for direct comparison. However, Wiebe et al. (2005) present an annotation study where they mark textual spans for subjective expressions in a newspaper corpus. They report pairwise κ values for three annotators ranging between 0.72 - 0.84 for the sentence level subjective/objective judgments. Wiebe et al. (2005) mark subjective spans, and do not explicitly perform the sentence level labeling task. They calculate the sentence level κ values based on the existence of a subjective expression span in the sentence. Although the task definitions, approaches and the corpora have quite disparate characteristics in both studies, we obtain comparable results when we merge *opinionated* and *polar_fact* categories.

4.3 Expression Level Agreement

At the expression level, annotators focus only on the sentences which were labeled as *opinionated* or *polar_fact* by both annotators. Annotators were instructed to mark text spans, and then, assign them the annotation types such as *polar_target, opinionexpression* etc. (see Figure 2). For calculating the text span agreement, we use the agreement metric presented by Wiebe et al. (2005) and Somasundaran et al. (2008). This metric corresponds to the precision (P) and recall (R) metrics in information retrieval where the decisions of one annotator are treated as the system; the decisions of the other annotator are treated as the gold standard; and the overlapping spans correspond to the correctly retrieved documents.

Somasundaran et al. (2008) present a discourse level annotation study in which opinion and target spans are marked and linked with each other in a meeting transcript corpus. Following Somasundaran et al. (2008), we compute three different measures for the text span agreement: *(i) exact matching* in which the text spans should perfectly match; *(ii) lenient (relaxed) matching* in which the overlap between spans is considered as a match, and *(iii) subset matching* in which a span has to be contained in another span in order to be considered as a match.[8] Agreement naturally increases as we relax the matching constraints. However, there were no differences between the *lenient* and the *subset* agreement values. Therefore, we report only the *exact* and *lenient matching* agreement results for each annotation type in Table 3. The same agreement results for the *lenient* and *subset* matching indicates that inexact matches are still very similar to each other, i.e., at least one span is totally contained in the other.

Somasundaran et al. (2008) do not report any F-measure. However, they report span agreement results in terms of precision and recall ranging between 0.44 - 0.87 for opinion spans and between 0.74 - 0.90 for the target spans. Wiebe et al. (2005) use the *lenient matching* approach for reporting text span agreements ranging between 0.59 - 0.81 for subjective expressions. We obtain higher agreement values for both opinion expression and target spans. We attribute this to the fact that the annotators look for opinion expression and target spans within the opinionated sentences which they agreed upon. Sentence level analysis indeed increases the reliability at the expression level. Compared to the high agreement on marking *target* spans, we obtain lower agreement values on marking *polar_target* spans. We observe that it is easier to attribute explicit expressions of evaluations to topic relevant entities compared to attributing evaluations implied by experiences to specific topic relevant entities in the reviews.

We calculated the agreement on identifying anaphoric references using the method introduced in (Passonneau, 2004) which utilizes Krippendorf's α (Krippendorff, 2004) for computing reliability for coreference annotation. We considered the overlapping *target* and *polar_target* spans together in this calculation, and obtained an α value of 0.29. Compared to Passonneau (α values from 0.46 to 0.74), we obtain a much lower agreement value. This may be due to the different definitions and organizations of the annotation tasks. Passonneau requires prior marking of all noun phrases (or instances which needs to be processed by the an-

[8] An example of *subset matching*: waste of time vs. total waste of time

Span	Exact			Lenient		
	P	R	F	P	R	F
opinionexpression	0.70	0.80	0.75	0.82	0.93	0.87
modifier	0.80	0.82	0.81	0.86	0.86	0.86
target	0.80	0.81	0.80	0.91	0.90	0.91
holder	0.75	0.72	0.73	0.93	0.88	0.91
polar_target	0.67	0.42	0.51	0.75	0.49	0.59

Table 3: Inter-annotator agreement on text spans at the expression level

notator). Annotator's task is to identify whether an instance refers to another marked entity in the discourse, and then, to identify corefering entity chains. However, in our annotation process annotators were tasked to identify only one entity as the referent, and was free to choose it from anywhere in the discourse. In other words, our chains contain only one entity. It is possible that both annotators performed correct resolutions, but still did not overlap with each other, as they resolve to different instances of the same entity in the discourse. We plan to further investigate reference resolution annotation discrepancies and perform corrections in the future.

Some annotation types require additional attributes to be labeled after marking the span. For instance, upon marking a text span as a *polar_target* or an *opinionexpression*, one has to label the *polarity* and *strength*. We consider the overlapping spans for each annotation type and use κ for reporting the agreement on these attributes. Table 4 shows the κ values.

Attribute	Markables	Agr.	κ
polarity	329	0.97	0.94
strength	329	0.74	0.55
modifier	136	0.88	0.77
polar_target_polarity	63	0.80	0.67

Table 4: Inter-annotator agreement at the expression level

We observe that the *strength* of the *opinionexpression* and the *polar_target_polarity* cannot be labeled as reliably as the *polarity* of the *opinionexpression*. 61% of the agreed upon *polar_targets* were labeled as negative by both annotators. On the other hand, only 35% of the agreed upon *opinionexpressions* were labeled as negative by both annotators. There were no neutral instances. This indicates that reviewers tend to report negative experiences using polar facts, probably objectively describing what has happened, but report positive experiences with explicit opinion expressions. Distribution of the *strength* attribute was as follows: *weak* 6%, *average* 54%, and *strong* 40%. The majority of the modifiers were annotated as intensifiers (70%), while 20% of the modifiers were labeled as negation.

4.4 Discussion

We analyzed the discrepancies in the annotations to gain insights about the challenges involved in various opinion related labeling tasks. At the sentence level, there were several trivial cases of disagreement, for instance, failing to recognize topic relevancy when the topic was not mentioned or referenced explicitly in the sentence, as in (18). Occasionally, annotators disagreed about whether a sentence that was written as a reaction to the other reviewers, as in (19), should be considered as topic relevant or not. Another source of disagreement included sentences similar to (20) and (21). One annotator interpreted them as universally true statements regardless of the topic, while the other attributed them to the discussed topic.

(18) Go to a state university if you know whats good for you!
(19) Those with sour grapes couldnt cut it, have an ax to grind, and are devoting their time to smearing the school.
(20) As far as learning, you really have to WANT to learn the material.
(21) On an aside, this type of education is not for the undisciplined learner.

Annotators easily distinguished the evaluations at the sentence level. However, they had difficulties distinguishing between a *polar_fact* and an opinion. For instance, both annotators agreed that the sentences (22) and (23) contain evaluations regarding the topic of the review. However, one annotator interpreted both sentences as objectively verifiable facts giving a positive impression about the school, while the other one treated them as opinions.

(22) All this work in the first 2 Years!
(23) The school has a reputation for making students work really hard.

Sentence level annotation increases the reliability of the expression level annotation in terms of marking text spans. However, annotators often had disagreements on labeling the *strength* attribute. For instance, one annotator labeled the

opinion expression in (24) as *strong*, while the other one labeled it as *average*. We observe that it is not easy to identify trivial causes of disagreements regarding *strength* as its perception by each individual is highly subjective. However, most of the disagreements occurred between *weak* and *average* cases.

(24) the experience that i have when i visit student finance is much ***like going to the dentist****, except when i leave, nothing is ever fixed.*

We did not apply any consolidation steps during our agreement studies. However, a final version of the corpus will be produced by the third judge (one of the co-authors) by consolidating the judgements of the two annotators.

5 Conclusions

We presented a corpus of consumer reviews from the *rateitall* and *eopinions* websites annotated with opinion related information. Existing opinion annotated user-generated corpora suffer from several limitations which result in difficulties for interpreting the experimental results and for performing error analysis. To name a few, they do not explicitly link the functional components of the opinions like targets, holders, or modifiers with the opinion expression; some of them do not mark opinion expression spans, none of them resolves anaphoric references in discourse. Therefore, we introduced a two level annotation scheme consisting of the sentence and expression levels, which overcomes the limitations of the existing review corpora. The sentence level annotation labels sentences for *(i)* relevancy to a given topic, and *(ii)* expressing an evaluation about the topic. Similar to (Wilson, 2008a), our annotation scheme allows capturing evaluations made with factual (objective) sentences. The expression level annotation further investigates on-topic sentences containing evaluations for pinpointing the properties (polarity, strength), and marking the functional components of the evaluations (opinion terms, modifiers, targets and holders), and linking them within a discourse. We applied the annotation scheme to the consumer review genre and presented an extensive inter-annotator study providing insights to the challenges involved in various opinion related labeling tasks in consumer reviews. Similar to the MPQA scheme, which is successfully applied to the newspaper genre, the annotation scheme treats opinions and evaluations as a composition of functional components and it is easily extendable. Therefore, we hypothesize that the scheme can also be applied to other genres with minor extensions or as it is. Finally, the corpus and the annotation manual will be made available at `http://www.ukp.tu-darmstadt.de/research/data/sentiment-analysis`.

Acknowledgements

This research was funded partially by the German Federal Ministry of Economy and Technology under grant 01MQ07012 and partially by the German Research Foundation (DFG) as part of the *Research Training Group on Feedback Based Quality Management in eLearning* under grant 1223. We are very grateful to Sandra Kübler for her help in organizing the annotators, and to Lizhen Qu for his programming support in harvesting the data.

References

Nicholas Asher, Farah Benamara, and Yvette Yannick Mathieu. 2008. Distilling opinion in discourse: A preliminary study. In *Coling 2008: Companion volume: Posters*, pages 7–10, Manchester, UK.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 2683–2688, Hyderabad, India.

Xiwen Cheng and Feiyu Xu. 2008. Fine-grained opinion topic and polarity identification. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 2710–2714, Marrekech, Morocco.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 355–362, Morristown, NJ, USA.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008*, pages 231–240, Palo Alto, California, USA.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 417–422, Genova, Italy.

Angela Fahrni and Manfred Klenner. 2008. Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proceedings of the Symposium on Affective Language in Human and Machine, AISB 2008 Convention*, pages 60 – 63, Aberdeen, Scotland.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD'04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177, Seattle, Washington.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text at the joint COLING-ACL Conference*, pages 1–8, Sydney, Australia.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methology*. Sage Publications, Thousand Oaks, California.

Rebecca J. Passonneau. 2004. Computing reliability for coreference. In *Proceedings of LREC*, volume 4, pages 1503–1506, Lisbon.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, New York.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *EMNLP-03: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 105–112.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2008. Discourse level opinion relations: An annotation study. In *In Proceedings of SIGdial Workshop on Discourse and Dialogue*, pages 129–137, Columbus, Ohio.

Veselin Stoyanov and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 817–824, Manchester, UK.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210.

Theresa Wilson and Janyce Wiebe. 2005. Annotating attributions and private states. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 53–60, Ann Arbor, Michigan.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada.

Theresa Wilson. 2008a. Annotating subjective content in meetings. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Theresa Ann Wilson. 2008b. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of Private States.* Ph.D. thesis, University of Pittsburgh.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50, Arlington, Virginia, USA.

Generating Focused Topic-specific Sentiment Lexicons

Valentin Jijkoun **Maarten de Rijke** **Wouter Weerkamp**
ISLA, University of Amsterdam, The Netherlands
jijkoun,derijke,w.weerkamp@uva.nl

Abstract

We present a method for automatically generating focused and accurate topic-specific subjectivity lexicons from a general purpose polarity lexicon that allow users to pin-point subjective on-topic information in a set of relevant documents. We motivate the need for such lexicons in the field of media analysis, describe a bootstrapping method for generating a topic-specific lexicon from a general purpose polarity lexicon, and evaluate the quality of the generated lexicons both manually and using a TREC Blog track test set for opinionated blog post retrieval. Although the generated lexicons can be an order of magnitude more selective than the general purpose lexicon, they maintain, or even improve, the performance of an opinion retrieval system.

1 Introduction

In the area of *media analysis*, one of the key tasks is collecting detailed information about opinions and attitudes toward specific topics from various sources, both offline (traditional newspapers, archives) and online (news sites, blogs, forums). Specifically, media analysis concerns the following system task: given a topic and list of documents (discussing the topic), find all instances of attitudes toward the topic (e.g., positive/negative sentiments, or, if the topic is an organization or person, support/criticism of this entity). For every such instance, one should identify the source of the sentiment, the polarity and, possibly, subtopics that this attitude relates to (e.g., specific targets of criticism or support). Subsequently, a (human) media analyst must be able to aggregate the extracted information by source, polarity or subtopics, allowing him to build support/criticism networks etc. (Altheide, 1996). Recent advances in language technology, especially in *sentiment analysis*, promise to (partially) automate this task.

Sentiment analysis is often considered in the context of the following two tasks:

- *sentiment extraction*: given a set of textual documents, identify phrases, clauses, sentences or entire documents that express attitudes, and determine the polarity of these attitudes (Kim and Hovy, 2004); and
- *sentiment retrieval*: given a topic (and possibly, a list of documents relevant to the topic), identify documents that express attitudes *toward this topic* (Ounis et al., 2007).

How can technology developed for sentiment analysis be applied to media analysis? In order to use a *sentiment extraction* system for a media analysis problem, a system would have to be able to determine which of the extracted sentiments are actually relevant, i.e., it would not only have to identify specific targets of all extracted sentiments, but also decide which of the targets are relevant for the topic at hand. This is a difficult task, as the relation between a *topic* (e.g., a movie) and specific targets of sentiments (e.g., acting or special effects in the movie) is not always straightforward, in the face of ubiquitous complex linguistic phenomena such as referential expressions ("... this beautifully shot *documentary*") or bridging anaphora ("the *director* did an excellent jobs").

In *sentiment retrieval*, on the other hand, the topic is initially present in the task definition, but it is left to the user to identify sources and targets of sentiments, as systems typically return a list of documents ranked by relevance and opinionatedness. To use a traditional sentiment retrieval system in media analysis, one would still have to manually go through ranked lists of documents returned by the system.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 585–594,
Uppsala, Sweden, 11-16 July 2010.

To be able to support media analysis, we need to combine the specificity of (phrase- or word-level) sentiment analysis with the topicality provided by sentiment retrieval. Moreover, we should be able to identify sources and specific targets of opinions.

Another important issue in the media analysis context is *evidence* for a system's decision. If the output of a system is to be used to inform actions, the system should present evidence, e.g., highlighting words or phrases that indicate a specific attitude. Most modern approaches to sentiment analysis, however, use various flavors of classification, where decisions (typically) come with confidence scores, but without explicit support.

In order to move towards the requirements of media analysis, in this paper we focus on two of the problems identified above: (1) pinpointing evidence for a system's decisions about the presence of sentiment in text, and (2) identifying specific targets of sentiment.

We address these problems by introducing a special type of lexical resource: a topic-specific subjectivity lexicon that indicates specific relevant targets for which sentiments may be expressed; for a given topic, such a lexicon consists of pairs (*syntactic clue*, *target*). We present a method for automatically generating a topic-specific lexicon for a given topic and query-biased set of documents. We evaluate the quality of the lexicon both manually and in the setting of an opinionated blog post retrieval task. We demonstrate that such a lexicon is highly *focused*, allowing one to effectively pinpoint evidence for sentiment, while being competetive with traditional subjectivity lexicons consisting of (a large number of) clue words.

Unlike other methods for topic-specific sentiment analysis, we do not expand a seed lexicon. Instead, we make an existing lexicon more focused, so that it can be used to actually pin-point subjectivity in documents relevant to a given topic.

2 Related Work

Much work has been done in sentiment analysis. We discuss related work in four parts: sentiment analysis in general, domain- and target-specific sentiment analysis, product review mining and sentiment retrieval.

2.1 Sentiment analysis

Sentiment analysis is often seen as two separate steps for determining subjectivity and polarity. Most approaches first try to identify subjective units (documents, sentences), and for each of these determine whether it is positive or negative. Kim and Hovy (2004) select candidate sentiment sentences and use word-based sentiment classifiers to classify unseen words into a negative or positive class. First, the lexicon is constructed from WordNet: from several seed words, the structure of WordNet is used to expand this seed to a full lexicon. Next, this lexicon is used to measure the distance between unseen words and words in the positive and negative classes. Based on word sentiments, a decision is made at the sentence level.

A similar approach is taken by Wilson et al. (2005): a classifier is learnt that distinguishes between polar and neutral sentences, based on a prior polarity lexicon and an annotated corpus. Among the features used are syntactic features. After this initial step, the sentiment sentences are classified as negative or positive; again, a prior polarity lexicon and syntactic features are used. The authors later explored the difference between prior and contextual polarity (Wilson et al., 2009): words that lose polarity in context, or whose polarity is reversed because of context.

Riloff and Wiebe (2003) describe a bootstrapping method to learn subjective extraction patterns that match specific syntactic templates, using a high-precision sentence-level subjectivity classifier and a large unannotated corpus. In our method, we bootstrap from a subjectivity lexicion rather than a classifier, and perform a topic-specific analysis, learning indicators of subjectivity toward a specific topic.

2.2 Domain- and target-specific sentiment

The way authors express their attitudes varies with the domain: An unpredictable movie can be positive, but unpredictable politicians are usually something negative. Since it is unrealistic to construct sentiment lexicons, or manually annotate text for learning, for every imaginable domain or topic, automatic methods have been developed.

Godbole et al. (2007) aim at measuring overall subjectivity or polarity towards a certain entity; they identify sentiments using domain-specific lexicons. The lexicons are generated from manually selected seeds for a broad domain such as *Health* or *Business*, following an approach similar to (Kim and Hovy, 2004). All named entites in a sentence containing a clue from a lexicon are

considered targets of sentiment for counting. Because of the data volume, no expensive linguistic processing is performed.

Choi et al. (2009) advocate a joint topic-sentiment analysis. They identify "sentiment topics," noun phrases assumed to be linked to a sentiment clue in the same expression. They address two tasks: identifying sentiment clues, and classifying sentences into positive, negative, or neutral. They start by selecting initial clues from SentiWordNet, based on sentences with known polarity. Next, the sentiment topics are identified, and based on these sentiment topics and the current list of clues, new potential clues are extracted. The clues can be used to classifiy sentences.

Fahrni and Klenner (2008) identify potential targets in a given domain, and create a target-specific polarity adjective lexicon. To this end, they find targets using Wikipedia, and associated adjectives. Next, the target-specific polarity of adjectives is detemined using Hearst-like patterns.

Kanayama and Nasukawa (2006) introduce polar atoms: minimal human-understandable syntactic structures that specify polarity of clauses. The goal is to learn new domain-specific polar atoms, but these are not target-specific. They use manually-created syntactic patterns to identify atoms and coherency to determine polarity.

In contrast to much of the work in the literature, we need to specialize subjectivity lexicons not for a domain and target, but for "topics."

2.3 Product features and opinions

Much work has been carried out for the task of mining product reviews, where the goal is to identify features of specific products (such as *picture*, *zoom*, *size*, *weight* for digital cameras) and opinions about these specific features in user reviews. Liu et al. (2005) describe a system that identifies such features via rules learned from a manually annotated corpus of reviews; opinions on features are extracted from the structure of reviews (which explicitly separate positive and negative opinions).

Popescu and Etzioni (2005) present a method that identifies product features for using corpus statistics, WordNet relations and morphological cues. Opinions about the features are extracted using a hand-crafted set of syntactic rules.

Targets extracted in our method for a topic are similar to features extracted in review mining for products. However, topics in our setting go beyond concrete products, and the diversity and generality of possible topics makes it difficult to apply such supervised or thesaurus-based methods to identify opinion targets. Moreover, in our method we directly use associations between targets and opinions to extract both.

2.4 Sentiment retrieval

At TREC, the Text REtrieval Conference, there has been interest in a specific type of sentiment analysis: opinion retrieval. This interest materialized in 2006 (Ounis et al., 2007), with the opinionated blog post retrieval task. Finding blog posts that are not just about a topic, but also contain an opinion on the topic, proves to be a difficult task. Performance on the opinion-finding task is dominated by performance on the underlying document retrieval task (the topical baseline).

Opinion finding is often approached as a two-stage problem: (1) identify documents relevant to the query, (2) identify opinions. In stage (2) one commonly uses either a binary classifier to distinguish between opinionated and non-opinionated documents or applies reranking of the initial result list using some opinion score. Opinion add-ons show only slight improvements over relevance-only baselines.

The best performing opinion finding system at TREC 2008 is a two-stage approach using reranking in stage (2) (Lee et al., 2008). The authors use SentiWordNet and a corpus-derived lexicon to construct an opinion score for each post in an initial ranking of blog posts. This opinion score is combined with the relevance score, and posts are reranked according to this new score. We detail this approach in Section 6. Later, the authors use domain-specific opinion indicators (Na et al., 2009), like "interesting story" (movie review), and "light" (notebook review). This domain-specific lexicon is constructed using feedback-style learning: retrieve an initial list of documents and use the top documents as training data to learn an opinion lexicon. Opinion scores per document are then computed as an average of opinion scores over all its words. Results show slight improvements (+3%) on mean average precision.

3 Generating Topic-Specific Lexicons

In this section we describe how we generate a lexicon of subjectivity clues and targets for a given *topic* and a list of *relevant documents* (e.g., re-

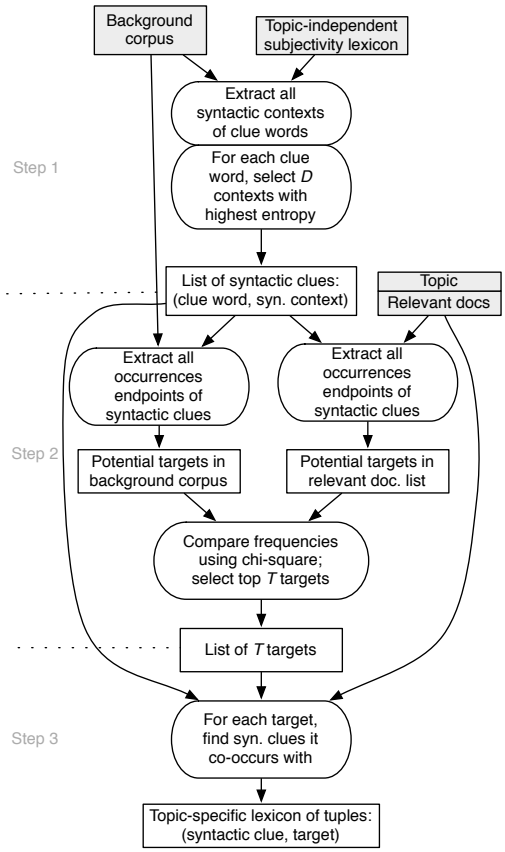


Figure 1: Our method for learning a topic-dependent subjectivity lexicon.

trieved by a search engine for the topic). As an additional resource, we use a large background corpus of text documents of a similar style but with diverse subjects; we assume that the relevant documents are part of this corpus as well. As the background corpus, we used the set of documents from the assessment pools of TREC 2006–2008 opinion retrieval tasks (described in detail in section 4). We use the Stanford lexicalized parser[1] to extract labeled dependency triples (*head, label, modifier*). In the extracted triples, all words indicate their category (*noun, adjective, verb, adverb,* etc.) and are normalized to lemmas.

Figure 1 provides an overview of our method; below we describe it in more detail.

3.1 Step 1: Extracting syntactic contexts

We start with a general domain-independent prior polarity lexicon of 8,821 clue words (Wilson et al., 2005). First, we identify *syntactic contexts* in which specific clue words can be used to express attitude: we try to find how a clue word can be syntactically linked to targets of sentiments. We take a simple definition of the syntactic context: a single labeled directed dependency relation. For every clue word, we extract all syntactic contexts, i.e., all dependencies, in which the word is involved (as head or as modifier) in the background corpus, along with their endpoints. Table 1 shows examples of clue words and contexts that indicate sentiments. For every clue, we only select those contexts that exhibit a high entropy among the lemmas at the other endpoint of the dependencies. E.g., in our background corpus, the verb *to like* occurs 97,179 times with a nominal subject and 52,904 times with a direct object; however, the entropy of lemmas of the subjects is 4.33, compared to 9.56 for the direct objects. In other words, subjects of *like* are more "predictable." Indeed, the pronoun *I* accounts for 50% of subjects, followed by *you* (14%), *they* (4%), *we* (4%) and *people* (2%). The most frequent objects of *like* are *it* (12%), *what* (4%), *idea* (2%), *they* (2%). Thus, objects of *to like* will be preferred by the method.

Our entropy-driven selection of syntactic contexts of a clue word is based on the following assumption:

> *Assumption 1:* In text, targets of sentiments are more diverse than sources of sentiments or other accompanying attributes such as location, time, manner, etc. Therefore targets exhibit higher entropy than other attributes.

For every clue word, we select the top D syntactic contexts whose entropy is at least half of the maximum entropy for this clue.

To summarize, at the end of Step 1 of our method, we have extracted a list of pairs (*clue word, syntactic context*) such that for occurrences of the clue word, the words at the endpoint of the syntactic dependency are likely to be targets of sentiments. We call such a pair a *syntactic clue*.

3.2 Step 2: Selecting potential targets

Here, we use the extracted syntantic clues to identify words that are likely to serve as specific targets for opinions about the topic in the relevant documents. In this work we only consider individual words as potential targets and leave exploring other options (e.g., NPs and VPs as targets) for future work. In extracting targets, we rely on the following assumption:

[1] http://nlp.stanford.edu/software/lex-parser.shtml

Clue word	Syntactic context	Target	Example
to like	has direct object	*u2*	*I do still like U2 very much*
to like	has clausal complement	*criticize*	*I don't like to criticize our intelligence services*
to like	has *about*-modifier	*olympics*	*That's what I like about Winter Olympics*
terrible	is adjectival modifier of	*idea*	*it's a terrible idea to recall judges for...*
terrible	has nominal subject	*shirt*	*And Neil, that shirt is terrible!*
terrible	has clausal complement	*can*	*It is terrible that a small group of extremists can ...*

Table 1: Examples of subjective syntactic contexts of clue words (based on Stanford dependencies).

Assumption 2: The list of relevant documents contains a substantial number of documents on the topic which, moreover, contain sentiments about the topic.

We extract all endpoints of all occurrences of the syntactic clues in the relevant documents, as well as in the background corpus. To identify potential attitude targets in the relevant documents, we compare their frequency in the relevant documents to the frequency in the background corpus using the standard χ^2 statistics. This technique is based on the following assumption:

Assumption 3: Sentiment targets related to the topic occur more often in subjective context in the set of relevant documents, than in the background corpus. In other words, while the background corpus contains sentiments towards very diverse subjects, the relevant documents tend to express attitudes related to the topic.

For every potential target, we compute the χ^2-score and select the top T highest scoring targets.

As the result of Steps 1 and 2, as candidate targets for a given topic, we only select words that occur in subjective contexts, and that do so more often than we would normally expect. Table 2 shows examples of extracted targets for three TREC topics (see below for a description of our experimental data).

3.3 Step 3: Generating topic-specific lexicons

In the last step of the method, we combine clues and targets. For each target identified in Step 2, we take all syntactic clues extracted in Step 1 that co-occur with the target in the relevant documents. The resulting list of triples (*clue word, syntactic context, target*) constitute the lexicon. We conjecture that an occurrence of a lexicon entry in a text indicates, with reasonable confidence, a subjective attitude towards the target.

Topic "Relationship between Abramoff and Bush"
abramoff lobbyist scandal fundraiser bush fund-raiser republican prosecutor tribe swirl corrupt corruption norquist democrat lobbying investigation scanlon reid lawmaker dealings president
Topic "MacBook Pro"
macbook laptop powerbook connector mac processor notebook fw800 spec firewire imac pro machine apple powerbooks ibook ghz g4 ata binary keynote drive modem
Topic: "Super Bowl ads"
ad bowl commercial fridge caveman xl endorsement advertising spot advertiser game super essential celebrity payoff marketing publicity brand advertise watch viewer tv football venue

Table 2: Examples of targets extracted at Step 2.

4 Data and Experimental Setup

We consider two types of evaluation. In the next section, we examine the quality of the lexicons we generate. In the section after that we evaluate lexicons quantitatively using the TREC Blog track benchmark.

For extrinsic evaluation we apply our lexicon generation method to a collection of documents containing opinionated utterances: blog posts. The Blogs06 collection (Macdonald and Ounis, 2006) is a crawl of blog posts from 100,649 blogs over a period of 11 weeks (06/12/2005–21/02/2006), with 3,215,171 posts in total. Before indexing the collection, we perform two preprocessing steps: (i) when extracting plain text from HTML, we only keep block-level elements longer than 15 words (to remove boilerplate material), and (ii) we remove non-English posts using TextCat[2] for language detection. This leaves us with 2,574,356 posts with 506 words per post on average. We index the collection using Indri,[3] version 2.10.

TREC 2006–2008 came with the task of *opinionated blog post retrieval* (Ounis et al., 2007). For each year a set of 50 topics was created, giv-

[2] http://odur.let.rug.nl/~vannoord/TextCat/

[3] http://www.lemurproject.org/indri/

ing us 150 topics in total. Every topic comes with a set of relevance judgments: Given a topic, a blog post can be either (i) nonrelevant, (ii) relevant, but not opinionated, or (iii) relevant and opinionated. TREC topics consist of three fields (*title*, *description*, and *narrative*), of which we only use the *title* field: a query of 1–3 keywords.

We use standard TREC evaluation measures for opinion retrieval: MAP (mean average precision), R-precision (precision within the top R retrieved documents, where R is the number of known relevant documents in the collection), MRR (mean reciprocal rank), P@10 and P@100 (precision within the top 10 and 100 retrieved documents). In the context of media analysis, recall-oriented measures such as MAP and R-precision are more meaningful than the other, early precision-oriented measures. Note that for the opinion retrieval task a document is considered relevant if it is on topic and contains opinions or sentiments towards the topic.

Throughout Section 6 below, we test for significant differences using a two-tailed paired t-test, and report on significant differences for $\alpha = 0.01$ ($^{\blacktriangle}$ and $^{\blacktriangledown}$), and $\alpha = 0.05$ ($^{\triangle}$ and $^{\triangledown}$).

For the quantative experiments in Section 6 we need a topical baseline: a set of blog posts potentially relevant to each topic. For this, we use the Indri retrieval engine, and apply the Markov Random Fields to model term dependencies in the query (Metzler and Croft, 2005) to improve topical retrieval. We retrieve the top 1,000 posts for each query.

5 Qualitative Analysis of Lexicons

Lexicon size (the number of entries) and selectivity (how often entries match in text) of the generated lexicons vary depending on the parameters D and T introduced above. The two rightmost columns of Table 4 show the lexicon size and the average number of matches per topic. Because our topic-specific lexicons consist of triples (*clue word, syntactic context, target*), they actually contain more words than topic-independent lexicons of the same size, but topic-specific entries are more selective, which makes the lexicon more focused. Table 3 compares the application of topic-independent and topic-specific lexicons to on-topic blog text.

We manually performed an explorative error analysis on a small number of documents, annotated using the smallest lexicon in Table 4 for the topic "March of the Pinguins." We assigned 186 matches of lexicon entries in 30 documents into four classes:

There are some *tragic* moments like eggs freezing , and predators *snatching* the females and *little* ones-you know the whole *NATURE* thing ... but this movie is *awesome*	There are some tragic moments l ike eggs freezing , and predators snatching the females and little ones-you know the whole NATURE thing ... but this **movie** is *awesome*
Saturday was more errands, then spent the evening with Dad and Stepmum, and *finally* was *able* to see March of the Penguins, which was *wonderful*. Christmas Day was *lovely*, surrounded by family, *good* food and drink, and *little* L to play with.	Saturday was more errands, then spent the evening with Dad and Stepmum, and finally was able to see March of the **Penguins**, which was *wonderful*. Christmas Day was lovely, surrounded by family, good food and drink, and little L to play with.

Table 3: Posts with highlighted targets (bold) and subjectivity clues (blue) using topic-independent (left) and topic-specific (right) lexicons.

- REL: sentiment towards a relevant target;
- CONTEXT: sentiment towards a target that is irrelevant to the topic due to context (e.g., opinion about a target "film", but refering to a film different from the topic);
- IRREL: sentiment towards irrelevant target (e.g., "game" for a topic about a movie);
- NOSENT: no sentiment at all

In total only 8% of matches were manually classified as REL, with 62% classified as NOSENT, 23% as CONTEXT, and 6% as IRREL. On the other hand, among documents assessed as opionionated by TREC assessors, only 13% did not contain matches of the lexicon entries, compared to 27% of non-opinionated documents, which does indicate that our lexicon does attempt to separate non-opinionated documents from opinionated.

6 Quantitative Evaluation of Lexicons

In this section we assess the quality of the generated topic-specific lexicons numerically and extrinsically. To this end we deploy our lexicons to the task of opinionated blog post retrieval (Ounis et al., 2007). A commonly used approach to this task works in two stages: (1) identify topically relevant blog posts, and (2) classify these posts as being opinionated or not. In stage 2 the standard

approach is to rerank the results from stage 1, instead of doing actual binary classification. We take this approach, as it has shown good performance in the past TREC editions (Ounis et al., 2007) and is fairly straightforward to implement. We also explore another way of using the lexicon: as a source for query expansion (i.e., adding new terms to the original query) in Section 6.2. For all experiments we use the collection described in Section 4.

Our experiments have two goals: to compare the use of topic-independent and topic-specific lexicons for the opinionated post retrieval task, and to examine how different settings for the parameters of the lexicon generation affect the empirical quality.

6.1 Reranking using a lexicon

To rerank a list of posts retrieved for a given topic, we opt to use the method that showed best performance at TREC 2008. The approach taken by Lee et al. (2008) linearly combines a (topical) relevance score with an opinion score for each post. For the opinion score, terms from a (topic-independent) lexicon are matched against the post content, and weighted with the probability of term's subjectivity. Finally, the sum is normalized using the Okapi BM25 framework. The final opinion score S_{op} is computed as in Eq. 1:

$$S_{op}(D) = \frac{Opinion(D) \cdot (k_1 + 1)}{Opinion(D) + k_1 \cdot (1 - b + \frac{b \cdot |D|}{avgdl})}, \quad (1)$$

where k_1, and b are Okapi parameters (set to their default values $k_1 = 2.0$, and $b = 0.75$), $|D|$ is the length of document D, and $avgdl$ is the average document length in the collection. The opinion score $Opinion(D)$ is calculated using Eq. 2:

$$Opinion(D) = \sum_{w \in O} P(sub|w) \cdot n(w, D), \quad (2)$$

where O is the set of terms in the sentiment lexicon, $P(sub|w)$ indicates the probability of term w being subjective, and $n(w, D)$ is the number of times term w occurs in document D. The opinion scoring can weigh lexicon terms differently, using $P(sub|w)$; it normalizes scores to cancel out the effect of varying document sizes.

In our experiments we use the method described above, and plug in the MPQA polarity lexicon.[4] We compare the results of using this topic-independent lexicon to the topic-dependent lexicons our method generates, which are also plugged into the reranking of Lee et al. (2008).

In addition to using Okapi BM25 for opinion scoring, we also consider a simpler method. As we observed in Section 5, our topic-specific lexicons are more selective than the topic-independent lexicon, and a simple number of lexicon matches can give a good indication of opinionatedness of a document:

$$S_{op}(D) = \min(n(O, D), 10)/10, \quad (3)$$

where $n(O, D)$ is the number of matches of the term of sentiment lexicon O in document D.

6.1.1 Results and observations

There are several parameters that we can vary when generating a topic-specific lexicon and when using it for reranking:

D: the number of syntactic contexts per clue
T: the number of extracted targets
$S_{op}(D)$: the opinion scoring function.
α: the weight of the opinion score in the linear combination with the relevance score.

Note that α does not affect the lexicon creation, but only how the lexicon is used in reranking. Since we want to assess the quality of lexicons, not in the opinionated retrieval performance as such, we factor out α by selecting the best setting for each lexicon (including the topic-independent) and each evaluation measure.

In Table 4 we present the results of evaluation of several lexicons in the context of opinionated blog post retrieval.

First, we note that reranking using all lexicons in Table 4 significantly improves over the relevance-only baseline for all evaluation measures. When comparing topic-specific lexicons to the topic-independent one, most of the differences are not statistically significant, which is surprising given the fact that most topic-specific lexicons we evaluated are substantially smaller (see the two rightmost columns in the table). The smallest lexicon in Table 4 is seven times more selective than the general one, in terms of the number of lexicon matches per document.

The only evaluation measure where the topic-independent lexicon consistently outperforms topic-specific ones, is Mean Reciprocal Rank that depends on a single relevant opinionated document high in a ranking. A possible explanation

[4] http://www.cs.pitt.edu/mpqa/

Lexicon			MAP	R-prec	MRR	P@10	P@100	|lexicon|	hits per doc
no reranking			0.2966	0.3556	0.6750	0.4820	0.3666	—	—
topic-independent			0.3182	0.3776	**0.7714**	**0.5607**	0.3980	8,221	36.17
D	T	S_{op}							
3	50	count	0.3191	0.3769	0.7276$^{\triangledown}$	0.5547	0.3963	2,327	5.02
3	100	count	0.3191	0.3777	0.7416	0.5573	0.3971	3,977	8.58
5	50	count	0.3178	0.3775	0.7246$^{\triangledown}$	0.5560	0.3931	2,784	5.73
5	100	count	0.3178	0.3784	0.7316$^{\triangledown}$	0.5513	0.3961	4,910	10.06
all	50	count	0.3167	0.3753	0.7264$^{\triangledown}$	0.5520	0.3957	4,505	9.34
all	100	count	0.3146	0.3761	0.7283$^{\triangledown}$	0.5347$^{\triangledown}$	0.3955	8,217	16.72
all	50	okapi	0.3129	0.3713	0.7247$^{\blacktriangledown}$	0.5333$^{\triangledown}$	0.3833$^{\triangledown}$	4,505	9.34
all	100	okapi	0.3189	0.3755	0.7162$^{\blacktriangledown}$	0.5473	0.3921	8,217	16.72
all	200	okapi	**0.3229**$^{\blacktriangle}$	**0.3803**	0.7389	0.5547	**0.3987**	14,581	29.14

Table 4: Evaluation of topic-specific lexicons applied to the opinion retrieval task, compared to the topic-independent lexicon. The two rightmost columns show the number of lexicon entries (average per topic) and the number of matches of lexicon entries in blog posts (average for top 1,000 posts).

is that the large general lexicon easily finds a few "obviously subjective" posts (those with heavily used subjective words), but is not better at detecting less obvious ones, as indicated by the recall-oriented MAP and R-precision.

Interestingly, increasing the number of syntactic contexts considered for a clue word (parameter D) and the number of selected targets (parameter T) leads to substantially larger lexicons, but only gives marginal improvements when lexicons are used for opinion retrieval. This shows that our bootstrapping method is effective at filtering out non-relevant sentiment targets and syntactic clues.

The evaluation results also show that the choice of opinion scoring function (Okapi or raw counts) depends on the lexicon size: for smaller, more focused lexicons unnormalized counts are more effective. This also confirms our intuition that for small, focused lexicons simple presence of a sentiment clue in text is a good indication of subjectivity, while for larger lexicons an overall subjectivity scoring of texts has to be used, which can be hard to interpret for (media analysis) users.

6.2 Query expansion with lexicons

In this section we evaluate the quality of targets extracted as part of the lexicons by using them for query expansion. Query expansion is a commonly used technique in information retrieval, aimed at getting a better representation of the user's information need by adding terms to the original retrieval query; for user-generated content, selective query expansion has proved very beneficial (Weerkamp et al., 2009). We hypothesize that if our method manages to identify targets that correspond to issues, subtopics or features associated with the topic, the extracted targets should be good candidates for query expansion. The experiments described below test this hypothesis.

Run	MAP	P@10	MRR
Topical blog post retrieval			
Baseline	0.4086	0.7053	0.7984
Rel. models	0.4017$^{\triangledown}$	0.6867	0.7383$^{\blacktriangledown}$
Subj. targets	0.4190$^{\triangle}$	0.7373$^{\triangle}$	0.8470$^{\triangle}$
Opinion retrieval			
Baseline	0.2966	0.4820	0.6750
Rel. models	0.2841$^{\blacktriangledown}$	0.4467$^{\blacktriangledown}$	0.5479$^{\blacktriangledown}$
Subj. targets	0.3075	0.5227$^{\blacktriangle}$	0.7196

Table 5: Query expansion using relevance models and topic-specific subjectivity targets. Significance tested against the baseline.

For every test topic, we select the 20 top-scoring targets as expansion terms, and use Indri to return 1,000 most relevant documents for the expanded query. We evaluate the resulting ranking using both topical retrieval and opinionated retrieval measures. For the sake of comparison, we also implemented a well-known query expansion method based on Relevance Models (Lavrenko and Croft, 2001): this method has been shown to work well in many settings. Table 5 shows evaluation results for these two query expansion methods, compared to the baseline retrieval run.

The results show that on topical retrieval query expansion using targets significantly improves retrieval performance, while using relevance models actually hurts all evaluation measures. The failure of the latter expansion method can be attributed to the relatively large amount of noise in user-generated content, such as boilerplate

material, timestamps of blog posts, comments etc. (Weerkamp and de Rijke, 2008). Our method uses full syntactic parsing of the retrieved documents, which might substantially reduce the amount of noise since only (relatively) well-formed English sentences are used in lexicon generation.

For opinionated retrieval, target-based expansion also improves over the baseline, although the differences are only significant for P@10. The consistent improvement for topical retrieval suggests that a topic-specific lexicon can be used both for query expansion (as described in this section) and for opinion reranking (as described in Section 6.1). We leave this combination for future work.

7 Conclusions and Future Work

We have described a bootstrapping method for deriving a topic-specific lexicon from a general purpose polarity lexicon. We have evaluated the quality of generated lexicons both manually and using a TREC Blog track test set for opinionated blog post retrieval. Although the generated lexicons can be an order of magnitude more selective, they maintain, or even improve, the performance of an opinion retrieval system.

As to future work, we intend to combine our method with known methods for topic-specific lexicon *expansion* (our method is rather concerned with lexicon "restriction"). Existing sentence- or phrase-level (trained) sentiment classifiers can also be used easily: when collecting/counting targets we can weigh them by "prior" score provided by such classifiers. We also want to look at more complex syntactic patterns: Choi et al. (2009) report that many errors are due to exclusive use of unigrams. We would also like to extend potential opinion targets to include multi-word phrases (NPs and VPs), in addition to individual words. Finally, we do not identify polarity yet: this can be partially inherited from the initial lexicon and refined automatically via bootstrapping.

Acknowledgements

This research was supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, by the DuOMAn project carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments under project nr STE-09-12, and by the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.066.512, 612.061.814, 612.061.-815, 640.004.802.

References

Altheide, D. (1996). *Qualitative Media Analysis*. Sage.

Choi, Y., Kim, Y., and Myaeng, S.-H. (2009). Domain-specific sentiment analysis using contextual feature generation. In *TSA '09: Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 37–44, New York, NY, USA. ACM.

Fahrni, A. and Klenner, M. (2008). Old Wine or Warm Beer: Target-Specific Sentiment Analysis of Adjectives. In *Proc.of the Symposium on Affective Language in Human and Machine, AISB 2008 Convention, 1st-2nd April 2008. University of Aberdeen, Aberdeen, Scotland*, pages 60 – 63.

Godbole, N., Srinivasaiah, M., and Skiena, S. (2007). Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Kanayama, H. and Nasukawa, T. (2006). Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363, Morristown, NJ, USA. Association for Computational Linguistics.

Kim, S. and Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of COLING 2004*.

Lavrenko, V. and Croft, B. (2001). Relevance-based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*.

Lee, Y., Na, S.-H., Kim, J., Nam, S.-H., Jung, H.-Y., and Lee, J.-H. (2008). KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. In *Proceedings of TREC 2008*.

Liu, B., Hu, M., and Cheng, J. (2005). Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*.

Macdonald, C. and Ounis, I. (2006). The TREC Blogs06 collection: Creating and analysing a blog test collection. Technical Report TR-2006-224, Department of Computer Science, University of Glasgow.

Metzler, D. and Croft, W. B. (2005). A markov random feld model for term dependencies. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, pages 472–479, New York, NY, USA. ACM Press.

Na, S.-H., Lee, Y., Nam, S.-H., and Lee, J.-H. (2009). Improving opinion retrieval based on query-specific sentiment lexicon. In *ECIR '09: Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, pages 734–738, Berlin, Heidelberg. Springer-Verlag.

Ounis, I., Macdonald, C., de Rijke, M., Mishne, G., and Soboroff, I. (2007). Overview of the TREC 2006 blog track. In *The Fifteenth Text REtrieval Conference (TREC 2006)*. NIST.

Popescu, A.-M. and Etzioni, O. (2005). Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Riloff, E. and Wiebe, J. (2003). Learning extraction patterns

for subjective expressions. In *Proceedings of the 2003 Conference on Empirical methods in Natural Language Processing (EMNLP)*.

Weerkamp, W., Balog, K., and de Rijke, M. (2009). A generative blog post retrieval model that uses query expansion based on external collections. In *Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-ICNLP 2009)*, Singapore.

Weerkamp, W. and de Rijke, M. (2008). Credibility improves topical blog post retrieval. In *Proceedings of ACL-08: HLT*, page 923931, Columbus, Ohio. Association for Computational Linguistics, Association for Computational Linguistics.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, Morristown, NJ, USA. Association for Computational Linguistics.

Wilson, T., Wiebe, J., and Hoffmann, P. (2009). Recognizing contextual polarity: an exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433.

Evaluating Multilanguage-Comparability of Subjectivity Analysis Systems

Jungi Kim, Jin-Ji Li and Jong-Hyeok Lee
Division of Electrical and Computer Engineering
Pohang University of Science and Technology, Pohang, Republic of Korea
{yangpa,ljj,jhlee}@postech.ac.kr

Abstract

Subjectivity analysis is a rapidly growing field of study. Along with its applications to various NLP tasks, much work have put efforts into multilingual subjectivity learning from existing resources. Multilingual subjectivity analysis requires language-independent criteria for comparable outcomes across languages. This paper proposes to measure the multilanguage-comparability of subjectivity analysis tools, and provides meaningful comparisons of multilingual subjectivity analysis from various points of view.

1 Introduction

The field of NLP has seen a recent surge in the amount of research on subjectivity analysis. Along with its applications to various NLP tasks, there have been efforts made to extend the resources and tools created for the English language to other languages. These endeavors have been successful in constructing lexicons, annotated corpora, and tools for subjectivity analysis in multiple languages.

There are multilingual subjectivity analysis systems available that have been built to monitor and analyze various concerns and opinions on the Internet; among the better known are OASYS from the University of Maryland that analyzes opinions on topics from news article searches in multiple languages (Cesarano et al., 2007)[1] and TextMap, an entity search engine developed by Stony Brook University for sentiment analysis along with other functionalities (Bautin et al., 2008).[2] Though these systems currently rely on English analysis tools and a machine translation (MT) technology to translate other languages into English, up-to-date research provides various ways to analyze subjectivity in multilingual environments.

Given sentiment analysis systems in different languages, there are many situations when the analysis outcomes need to be multilanguage-comparable. For example, it has been common these days for the Internet users across the world to share their views and opinions on various topics including music, books, movies, and global affairs and incidents, and also multinational companies such as Apple and Samsung need to analyze customer feedbacks for their products and services from many countries in different languages. Governments may also be interested in monitoring terrorist web forums or its global reputation. Surveying these opinions and sentiments in various languages involves merging the analysis outcomes into a single database, thereby objectively comparing the result across languages.

If there exists an ideal subjectivity analysis system for each language, evaluating the multilanguage-comparability would be unnecessary because the analysis in each language would correctly identify the exact meanings of all input texts regardless of the language. However, this requirement is not fulfilled with current technology, thus the need for defining and measuring the multilanguage-comparability of subjectivity analysis systems is evident.

This paper proposes to evaluate the multilanguage-comparability of multilingual subjectivity analysis systems. We build a number of subjectivity classifiers that distinguishes subjective texts from objective ones, and measure the multilanguage-comparability according to our proposed evaluation method. Since subjectivity analysis tools in languages other than English are not readily available, we focus our experiments on comparing different methods to build multilingual analysis systems from the resources and systems

[1] http://oasys.umiacs.umd.edu/oasysnew/
[2] http://www.textmap.com/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 595–603,
Uppsala, Sweden, 11-16 July 2010.

created for English. These approaches enable us to extend a monolingual system to many languages with a number of freely available NLP resources and tools.

2 Related Work

Much research have been put into developing methods for multilingual subjectivity analysis recently. With the high availability of subjectivity resources and tools in English, an easy and straightforward approach would be to employ a machine translation (MT) system to translate input texts in target languages into English then carry out the analyses using an existing subjectivity analysis tool (Kim and Hovy, 2006; Bautin et al., 2008; Banea et al., 2008). Mihalcea et al. (2007) and Banea et al. (2008) proposed a number of approaches exploiting a bilingual dictionary, a parallel corpus, and an MT system to port the resources and systems available in English to languages with limited resources.

For subjectivity lexicons translation, Mihalcea et al. (2007) and Wan (2008) used the first sense in a bilingual dictionary, Kim and Hovy (2006) used a parallel corpus and a word alignment tool to extract translation pairs, and Kim et al. (2009) used a dictionary to translate and a link analysis algorithm to refine the matching intensity.

To overcome the shortcomings of available resources and to take advantage of ensemble systems, Wan (2008) and Wan (2009) explored methods for developing a hybrid system for Chinese using English and Chinese sentiment analyzers. Abbasi et al. (2008) and Boiy and Moens (2009) have created manually annotated gold standards in target languages and studied various feature selection and learning techniques in machine learning approaches to analyze sentiments in multilingual web documents.

For learning multilingual subjectivity, the literature tentatively concludes that translating lexicon is less dependable in terms of preserving subjectivity than corpus translation (Mihalcea et al., 2007; Wan, 2008), and though corpus translation results in modest performance degradation, it provides a viable approach because no manual labor is required (Banea et al., 2008; Brooke et al., 2009).

Based on the observation that the performances of subjectivity analysis systems in comparable experimental settings for two languages differ,

Texts with an identical negative sentiment:
* The iPad could cannibalize the e-reader market.
* 아이패드가(iPad) 전자책 시장을(e-reader market) 위축시킬 수 있다(could cannibalize).

Texts with different strengths of positive sentiments:
* Samsung cell phones have excellent battery life.
* 삼성(Samsung) 휴대전화(cell phone) 배터리는(battery) 그럭저럭(somehow or other) 오래간다(last long).

Figure 1: Examples of sentiments in multilingual text

Banea et al. (2008) have attributed the variations in the difficulty level of subjectivity learning to the differences in language construction. Bautin et al. (2008)'s system analyzes the sentiment scores of entities in multilingual news and blogs and adjusted the sentiment scores using entity sentiment probabilities of languages.

3 Multilanguage-Comparability

3.1 Motivation

The quality of a subjectivity analysis tool is measured by its ability to distinguish subjectivity from objectivity and/or positive sentiments from negative sentiments. Additionally, a multilingual subjectivity analysis system is required to generate unbiased analysis results across languages; the system should base its outcome solely on the subjective meanings of input texts irrespective of the language, and the equalities and inequalities of subjectivity labels and intensities must be useful within and throughout the languages.

Let us consider two cases where the pairs of multilingual inputs in English and Korean have identical and different subjectivity meanings (Figure 1). The first pair of texts carry a negative sentiment about how the release of a new electronics device might affect an emerging business market. When a multilanguage-comparable system is inputted with such a pair, its output should appropriately reflect the negative sentiment, and be identical for both texts. The second pair of texts share a similar positive sentiment about a mobile device's battery capacity but with different strengths. A good multilingual system must be able to identify the positive sentiments and distinguish the differences in their intensities.

However, these kinds of conditions cannot be measured with performance evaluations indepen-

dently carried out on each language; A system with a dissimilar ability to analyze subjective expressions from one language to another may deliver opposite labels or biased scores on texts with an identical subjective meaning, and vice versa, but still might produce similar performances on the evaluation data.

Macro evaluations on individual languages cannot provide any conclusions on the system's multilanguage-comparability capability. To measure how much of a system's judgment principles are preserved across languages, an evaluation from a different perspective is necessary.

3.2 Evaluation Approach

An evaluation of multilanguage-comparability may be done in two ways: measuring agreements in the outcomes of a pair of multilingual texts with an identical subjective meaning, or measuring the consistencies in the label and/or accordance in the order of intensity of a pair of texts with different subjectivities.

There are advantages and disadvantages to each approaches. The first approach requires multilingual texts aligned at the level of specificity, for instance, document, sentence and phrase, that the subjectivity analysis system works. Text corpora for MT evaluation such as newspapers, books, technical manuals, and government official records provide a wide variety of parallel texts, typically at the sentence level. Annotating these types of corpus can be efficient; as parallel texts must have identical semantic meanings, subjectivity–related annotations for one language can be projected into other languages without much loss of accuracy.

The latter approach accepts any pair of multilingual texts as long as they are annotated with labels and/or intensity. In this case, evaluating the label consistency of a multilingual system is only as difficult as evaluating that of a monolingual system; we can produce all possible pairs of texts from test corpora annotated with labels for each language. Evaluating with intensity is not easy for the latter approach; if test corpora already exist with intensity annotations for both languages, normalizing the intensity scores to a comparable scale is necessary (yet is uncertain unless every pair is checked manually), otherwise every pair of multilingual texts needs a manual annotation with its relative order of intensity.

In this paper, we utilize the first approach because it provides a more rational means; we can reasonably hypothesize that text translated into another language by a skilled translator carries an identical semantic meaning and thereby conveys identical subjectivity. Therefore the required resource is more easily attained in relatively inexpensive ways.

For evaluation, we measure the consistency in the subjectivity labels and the correlation of subjectivity intensity scores of parallel texts. Section 5.1 describes the details of evaluation metrics.

4 Multilingual Subjectivity System

We create a number of multilingual systems consisting of multiple subsystems each processing a language, where one system analyzes English, and the other systems analyze the Korean, Chinese, and Japanese languages. We try to reproduce a set of systems using diverse methods in order to compare the systems and find out which methods are more suitable for multilanguage-comparability.

4.1 Source Language System

We adopt the three systems described below as our source language systems: a state-of-the-art subjectivity classifier, a corpus-based, and a lexicon-based systems. The resources needed for developing the systems or the system itself are readily available for research purposes. In addition, these systems cover the general spectrum of current approaches to subjectivity analysis.

State-of-the-art (S-SA): OpinionFinder is a publicly-available NLP tool for subjectivity analysis (Wiebe and Riloff, 2005; Wilson et al., 2005).[3] The software and its resources have been widely used in the field of subjectivity analysis, and it has been the de facto standard system against which new systems are validated. We use a high-coverage classifier from the OpinionFinder's two sentence-level subjectivity classifiers. This Naive Bayes classifier builds upon a corpus annotated by a high-precision classifier with the bootstrapping of the corpus and extraction patterns. The classifier assesses a sentence's subjectivity with a label and a score for confidence in its judgment.

Corpus-based (S-CB): The MPQA opinion corpus is a collection of 535 newspaper articles in English annotated with opinions and private states at

[3] http://www.cs.pitt.edu/mpqa/opinionfinderrelease/, version 1.5

the sub-sentence level (Wiebe et al., 2003).[4] We retrieve the sentence level subjectivity labels for 11,111 sentences using the set of rules described in (Wiebe and Riloff, 2005). The corpus provides a relatively balanced corpus with 55% subjective sentences. We train an ML-based classifier using the corpus. Previous studies have found that, among several ML-based approaches, the SVM classifier generally performs well in many subjectivity analysis tasks (Pang et al., 2002; Banea et al., 2008).

We use SVM^{Light} with its default configurations,[5] inputted with a sentence represented as a feature vector of word unigrams and their counts in the sentence. An SVM score (a margin or the distance from a learned decision boundary) with a positive value predicts the input as being subjective, and negative value as objective.

Lexicon-based (S-LB): OpinionFinder contains a list of English subjectivity clue words with intensity labels (Wilson et al., 2005). The lexicon is compiled from several manually and automatically built resources and contains 6885 unique entries.

Riloff and Wiebe (2003) constructed a high-precision classifier for contiguous sentences using the number of strong and weak subjective words in current and nearby sentences. Unlike previous work, we do not (or rather, cannot) maintain assumptions about the proximity of input text. Using the lexicon, we build a simple and high-coverage rule-based subjectivity classifier. Setting the scores of strong and weak subjective words as 1.0 and 0.5, we evaluate the subjectivity of a given sentence as the sum of subjectivity scores; above a threshold, the input is subjective, and otherwise objective. The threshold value is optimized for an F-measure using the MPQA corpus, and is set to 1.0 throughout our experiments.

4.2 Target Language System

To construct a target language system leveraging on available resources in the source language, we consider three approaches from previous literature:

1. translating test sentences in target language into source language and inputting them into a source language system (Kim and Hovy, 2006; Bautin et al., 2008; Banea et al., 2008)
2. translating a source language training corpus into target language and creating a corpus-based system in target language (Banea et al., 2008)
3. translating a subjectivity lexicon from source language to target language and creating a lexicon-based system in target language (Mihalcea et al., 2007)

Each approach has its advantages and disadvantages. The advantage of the first approach is its simple architecture, clear separation of subjectivity and MT systems, and that it has only one subjectivity system, and is thus easier to maintain. Its disadvantage is that the time-consuming MT has to be executed for each text input. In the second and third approaches, a subjectivity system in the target language is constructed sharing corpora, rules, and/or features with the source language system. Later on, it may also include its own set of resources specifically engineered for the target language as a performance improvement. However, keeping the systems up-to-date would require as much effort as the number of languages. All three approaches use MT, and would suffer significantly if the translation results are poor.

Using the first approach, we can easily adopt all three source language systems;

- Target input translated into source, analyzed by source language system **S-SA**
- Target input translated into source, analyzed by source language system **S-CB**
- Target input translated into source, analyzed by source language system **S-LB**

The second and the third approaches are carried out as follows:

Corpus-based (T-CB): We translate the MPQA corpus into the target languages sentence by sentence using a web-based service.[6] Using the same method for **S-CB**, we train an SVM model for each language with the translated training corpora.

Lexicon-based (T-LB): This classifier is identical to **S-LB**, where the English lexicon is replaced by one of the target languages. We automatically translate the lexicon using free bilingual dictionaries.[7] First, the entries in the lexicon are looked

[4] http://www.cs.pitt.edu/mpqa/databaserelease/, version 1.2

[5] http://svmlight.joachims.org/, version 6.02

[6] Google Translate (http://translate.google.com/)

[7] quick_english-korean, quick_eng-zh_CN, and JMDict from StarDict (http://stardict.sourceforge.net/) licensed under GPL and EDRDG.

Table 1: Agreement on subjectivity (S for subjective, O objective) of 859 sentence chunks in Korean between two annotators (An. 1 and An. 2).

		An. 2		
		S	O	Total
An. 1	S	371	93	464
	O	23	372	395
	Total	394	465	859

Table 2: Agreement on projection of subjectivity (S for subjective, O objective) from Korean (KR) to English (EN) by one annotator.

		EN		
		S	O	Total
KR	S	458	6	464
	O	12	383	395
	Total	470	389	859

up in the dictionary, if they are found, we select the first word in the first sense of the definition. If the entry is not in the dictionary, we lemmatize it,[8] then repeat the search. Our simple approach produces moderate-sized lexicons (3,808, 3,980, 3,027 for Korean, Chinese, and Japanese) compared to Mihalcea et al. (2007)'s complicated translation approach (4,983 Romanian words). The threshold values are optimized using the MPQA corpus translated into each target language.[9]

5 Experiment

5.1 Experimental Setup

Test Corpus

Our evaluation corpus consists of 50 parallel newspaper articles from the Donga Daily News Website.[10] The website provides news articles in Korean and their human translations in English, Japanese, and Chinese. We selected articles that contain *Editorial* in its English title from a 30-day period. Three human annotators who are fluent in the two languages manually annotated N-to-N sentence alignments for each language pairs (KR-EN, KR-CH, KR-JP). By keeping only the sentence chunks whose Korean chunk appears in all language pairs, we were left with 859 sentence chunk pairs.

The corpus was preprocessed with NLP tools for each language,[11] and the Korean, Chinese, and Japanese texts were translated into English with the same web-based service used to translate the training corpus in Section 4.2.

Manual Annotation and Agreement Study

To assess the performance of our subjectivity analysis systems, the Korean sentence chunks were manually annotated by two native speakers of Korean with *Subjective* and *Objective* labels (Table 1). A proportion agreement of 0.86 and a kappa value of 0.73 indicate a substantial agreement between the two annotators. We set aside 743 sentence chunks that both annotators agreed on for the automatic evaluation of subjectivity analysis systems, thereby removing the borderline cases, which are difficult even for humans to assess. The corresponding sentence chunks for other languages were extracted and tagged with labels equivalent to Korean chunks.

In addition, to verify how consistently the subjectivity of the original texts is projected to the translated, we carried out another manual annotation and agreement study with Korean and English sentence chunks (Table 2).

Note that our cross-lingual agreement study is similar to the one carried out by Mihalcea et al. (2007), where two annotators labeled the sentence subjectivity of a parallel text in different languages. They reported that, similarly to monolingual annotations, most cases of disagreements on annotations are due to the differences in the annotators' judgments on subjectivity, and the rest from subjective meanings lost in the translation process and figurative language such as irony.

To avoid the role played by annotators' private views from disagreements, the subjectivity of sentence chunks in English were manually annotated by one of the annotators for the Korean text. Judged by the same annotator, we speculate that the disagreement in the annotation should account only for the inconsistency in the subjectivity projection. By proportion, the agreement between the annotation of Korean and English is 0.97, and the kappa is 0.96, suggesting an almost perfect agreement. Only a small number of sentence chunk pairs have inconsistent labels; six chunks in Ko-

[8] JWI (http://projects.csail.mit.edu/jwi/)

[9] Korean 1.0, Chinese 1.0, and Japanese 0.5

[10] http://www.donga.com/

[11] Stanford POS Tagger 1.5.1 and Stanford Chinese Word Segmenter 2008-05-21 (http://nlp.stanford.edu/software/), Chasen 2.4.4 (http://chasen-legacy.sourceforge.jp/), Korean Morphological Analyzer (KoMA) (http://kle.postech.ac.kr/)

Implicit sentiment expressed through translation:
* 시간이 갈수록(with time) 그 격차가(disparity/gap) 벌어지고 있다(widening).
* Worse, the (economic) disparity (between South Korea and North Korea) is worsening with time.

Sentiment lost in translation:
* 인도의 타타 자동차회사는(India's Tata Motors) 2200달러짜리 자동차 나노를(2,200-dollar automobile Nano) 내놓아(presented) 주목을 끌었다 (drew attention).
* India's Tata Motors has produced the 2,200-dollar subcompact Nano.

Figure 2: Excerpts from Donga Daily News with differing sentiments between parallel texts

rean lost subjectivity in translation, and implied subjective meanings in twelve chunks were expressed explicitly through interpretation. Excerpts from our corpus show two such cases (Figure 2).

Evaluation Metrics

To evaluate the multilanguage-comparability of subjectivity analysis systems, we measure 1) how consistently the system assigns subjectivity labels and 2) how closely numeric scores for systems' confidences correlate with regard to parallel texts in different languages.

In particular, we use Cohen's kappa coefficient for the first and Pearson's correlation coefficient for the latter. These widely used metrics provide useful comparability measures for categorical and quantitative data.

Both coefficients are scaled from -1 to $+1$, indicating negative to positive correlations. Kappa measures are corrected for chance, thereby yielding better measurements than agreement by proportion. The characteristics of Pearson's correlation coefficient that it measures linear relationships and is independent of change in origin, scale, and unit comply with our experiments.

5.2 Subjectivity Classification

Our multilingual subjectivity analysis systems were evaluated on the test corpora described in Section 5.1 (Table 3).

Due to the difference in testbeds, the performance of the state-of-the-art English system (**S-SA**) on our corpus is lower by about 10% relatively than the performance reported on the MPQA corpus.[12] However, it still performs sufficiently well and provides the most balanced results among the three source language systems; The corpus-based system (**S-CB**) classifies with a high precision, and the lexicon-based (**S-LB**) with a high recall. The source language systems (**S-SA,-CB,-LB**) lose a small percentage in precision when inputted with translations, but the recalls are generally on a par or even higher in the target languages.

For the systems created from target language resources, Corpus-based systems (**T-CB**) generally perform better than the ones with source language resource (**S-CB**), and lexicon-based systems (**T-LB**) perform worse than (**S-LB**). Similarly to systems with source language resources, **T-CB** classifies with a high precision and **T-LB** with a high recall, but the gap is less. Among the target languages, Korean tends to have a higher precision, and Japanese a higher recall than other languages in most systems.

Overall, **S-SA** provides easy accessibility when analyzing both the source and the target languages, with a balanced precision and recall performance. Among the other approaches, only **T-CB** is better in all measures than **S-SA**, and **S-LB** performs best on F-measure evaluations.

5.3 Multilanguage-Comparability

The evaluation results on multilanguage-comparability are presented in Table 4. The subjectivity analysis systems are evaluated with all language pairs with kappa and Pearson's correlation coefficients. Kappa and Pearson's correlation values are consistent with each other; Pearson's correlation between the two evaluation measures is 0.91.

We observe a distinct contrast in performances between corpus-based systems (**S-CB** and **T-CB**) and lexicon-based systems (**S-LB** and **T-LB**); All corpus-based systems show moderate agreements while agreements on lexicon-based systems are only fair.

Within corpus-based systems, **S-CB** performs better with language pairs that include English, and **T-CB** performs better with language pairs of the target languages.

For lexicon-based systems, systems in the target languages (**T-LB**) performs the worst with only slight to fair agreements between languages. Lexicon-based systems and state-of-the-art systems in the source language (**S-LB** and **S-SA**) result in average performances.

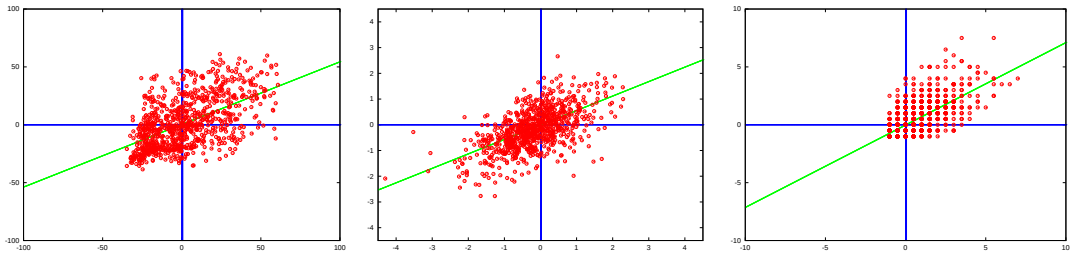
[12] precision, recall, and F-measure of 79.4, 70.6, and 74.7.

Table 3: Performance of subjectivity analysis with precision (P), recall (R), and F-measure (F). S-SA,-CB,-LB systems in Korean, Chinese, Japanese indicate English analysis systems inputted with translations of the target languages into English.

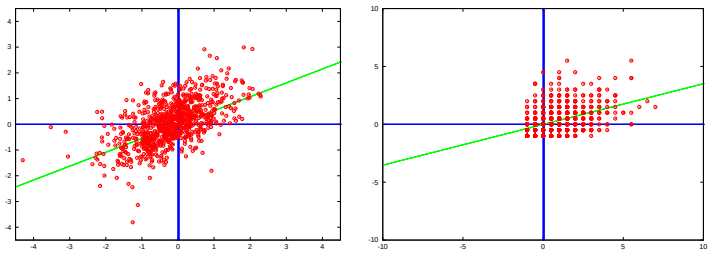
	English			Korean			Chinese			Japanese		
	P	R	F	P	R	F	P	R	F	P	R	F
S-SA	71.1	63.5	67.1	70.7	61.1	65.6	67.3	68.8	68.0	69.1	67.5	68.3
S-CB	**74.4**	53.9	62.5	**74.5**	52.2	61.4	71.1	63.3	67.0	**72.9**	65.3	68.9
S-LB	62.5	**87.7**	**73.0**	62.9	**87.7**	**73.3**	59.9	**91.5**	**72.4**	61.8	**94.1**	**74.6**
T-CB				72.4	67.5	69.8	**75.0**	66.2	70.3	72.5	70.3	71.4
T-LB				59.4	71.0	64.7	58.4	82.3	68.2	56.9	92.4	70.4

Table 4: Performance of multilanguage-comparability: kappa coefficient (κ) for measuring comparability of classification labels and Pearson's correlation coefficient (ρ) for classification scores for English (EN), Korean (KR), Chinese (CH), and Japanese (JP). Evaluations of T-CB,-LB for language pairs including English are carried out with results from S-CB,-LB for English and T-CB,-LB for target languages.

	S-SA		S-CB		S-LB		T-CB		T-LB	
	κ	ρ	κ	ρ	κ	ρ	κ	ρ	κ	ρ
EN & KR	0.41	0.55	**0.45**	**0.60**	0.37	0.59	0.42	**0.60**	0.25	0.41
EN & CH	0.39	0.54	**0.41**	**0.62**	0.33	0.52	0.39	0.57	0.22	0.38
EN & JP	0.39	0.53	**0.43**	**0.65**	0.30	0.59	0.40	0.59	0.15	0.33
KR & CH	0.36	0.54	0.39	0.59	0.28	0.57	**0.46**	**0.64**	0.23	0.37
KR & JP	0.37	0.60	0.44	0.69	0.50	0.69	**0.63**	**0.76**	0.18	0.38
CH & JP	0.37	0.53	**0.49**	**0.66**	0.29	0.57	0.46	0.63	0.22	0.46
Average	0.38	0.55	0.44	**0.64**	0.35	0.59	**0.46**	0.63	0.21	0.39



(a) S-SA (b) S-CB (c) S-LB



(d) T-CB (e) T-LB

Figure 3: Scatter plots of English (x-axis) and Korean (y-axis) subjectivity scores from state-of-the-art (S-SA), corpus-based (S-CB), and lexicon-based (S-LB) systems of the source language, and corpus-based with translated corpora (T-CB), and lexicon-based with translated lexicon (T-LB) systems. Slanted lines in figures are best-fit lines through the origins.

Figure 3 shows scatter plots of subjectivity scores of our English and Korean test corpora evaluated on different systems; the data points on the first and the third quadrants are occurrences of label agreements, and the second and the fourth are disagreements. Linearly scattered data points are more correlated regardless of the slope.

Figure 3a shows a moderate correlation for multilingual results from the state-of-the-art system (**S-SA**). Agreements on objective instances are clustered together while agreements on subjective instances are diffused over a wide region.

Agreements between the source language corpus-based system (**S-CB**) and the corpus-based system trained with translated resources (**T-CB**) are more distinctively correlated than the results for other pairs of systems (Figures 3b and 3d). We notice that **S-CB** seems to have a lower number of outliers than **T-CB**, but slightly more diffusive.

Lexicon-based systems (**S-LB**, **T-LB**) generate noticeably uncorrelated scores (Figures 3c and 3e). We observe that the results from the English system with translated inputs (**S-LB**) is more correlated than those from systems with translated lexicons (**T-LB**), and that analysis results from both systems are biased toward subjective scores.

6 Discussion

Which approach is most suitable for multilingual subjectivity analysis?

In our experiments, the corpus-based systems trained on corpora translated from English to the target languages (**T-CB**) perform well for subjectivity classification and multilanguage-comparability measures on the whole. However, the methods we employed to expand the languages were naively carried out without much considerations for optimization. Further adjustments could improve the other systems for both classification and multilanguage-comparability performances.

Is there a correlation between classification performance and multilanguage-comparability?

Lexicon-based systems in the source language (**S-LB**) have good overall classification performances, especially on recall and F-measures. However, these systems performs worse on multilanguage-comparability than other systems with poorer classification performances. Intrigued by the observation, we tried to measure which criteria for classification performance influences multilanguage-comparability. We again employed Pearson's correlation metrics to measure the correlations of precision (P), recall (R), and F-measures (F) to kappa (κ) and Pearson's correlation (ρ) values.

Specifically, we measure the correlations between the sums of P, the sums of R, and the sums of F to κ and ρ for all pairs of systems.[13] The correlations of P with κ and ρ are 0.78 and 0.68, R -0.38 and -0.28, and F -0.20 and -0.05. These numbers strongly suggest that multilanguage-comparability correlates with the precisions of classifiers.

However, we cannot always expect a high-precision multilingual subjectivity classifier to be multilanguage-comparable as well. For example, the **S-SA** system has a much higher precision than **S-LB** consistently over all languages, but their multilanguage-comparability performances differed only by small amounts.

7 Conclusion

Multilanguage-comparability is an analysis system's ability to retain its decision criteria across different languages. We implemented a number of previously proposed approaches to learning multilingual subjectivity, and evaluated the systems on multilanguage-comparability as well as classification performance. Our experimental results provide meaningful comparisons of the multilingual subjectivity analysis systems across various aspects.

Also, we developed a multilingual subjectivity evaluation corpus from a parallel text, and studied inter-annotator, inter-language agreements on subjectivity, and observed persistent subjectivity projections from one language to another from a parallel text.

For future work, we aim extend this work to constructing a multilingual sentiment analysis system and evaluate it with multilingual datasets such as product reviews collected from different countries. We also plan to resolve the lexicon-based classifiers' classification bias towards subjective meanings with a list of objective words (Esuli and Sebastiani, 2006) and their multilingual expansion (Kim et al., 2009), and evaluate the multilanguage-comparability of systems constructed with resources from different sources.

[13] Pairs of values such as 71.1 + 70.7 and 0.41 for precisions and Kappa of S-SA for English and Korean.

Acknowledgement

We thank the anonymous reviewers for valuable comments and helpful suggestions. This work is supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) (2009-0075211), and in part by the BK 21 project in 2010.

References

Ahmed Abbasi, Hsinchun Chen, and Arab Salem. 2008. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems*, 26(3):1–34.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 127–135, Morristown, NJ, USA.

Mikhail Bautin, Lohit Vijayarenu, and Steven Skiena. 2008. International sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multlingual Web texts. *Information Retrieval*, 12:526–558.

Julian Brooke, Milan Tofiloski, and Maite Taboada. 2009. Cross-linguistic sentiment analysis: From english to spanish. In *Proceedings of RANLP 2009*, Borovets, Bulgaria.

Carmine Cesarano, Antonio Picariello, Diego Reforgiato, and V.S. Subrahmanian. 2007. The oasys 2.0 opinion analysis system: A demo. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, pages 417–422, Geneva, IT.

Soo-Min Kim and Eduard Hovy. 2006. Identifying and analyzing judgment opinions. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT/NAACL'06)*, pages 200–207, New York, USA.

Jungi Kim, Hun-Young Jung, Sang-Hyob Nam, Yeha Lee, and Jong-Hyeok Lee. 2009. Found in translation: Conveying subjectivity of a lexicon of one language into another using a bilingual dictionary and a link analysis algorithm. In *ICCPOL '09: Proceedings of the 22nd International Conference on Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, pages 112–121, Berlin, Heidelberg.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 976–983, Prague, CZ.

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

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xiaojun Wan. 2008. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 553–561, Honolulu, Hawaii, October. Association for Computational Linguistics.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243, Suntec, Singapore, August. Association for Computational Linguistics.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005)*, pages 486–497, Mexico City, Mexico.

Janyce Wiebe, E. Breck, Christopher Buckley, Claire Cardie, P. Davis, B. Fraser, Diane Litman, D. Pierce, Ellen Riloff, Theresa Wilson, D. Day, and Mark Maybury. 2003. Recognizing and organizing opinions expressed in the world press. In *Proceedings of the 2003 AAAI Spring Symposium on New Directions in Question Answering*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP'05)*, pages 347–354, Vancouver, CA.

Error Detection for Statistical Machine Translation Using Linguistic Features

Deyi Xiong, Min Zhang, Haizhou Li
Human Language Technology
Institute for Infocomm Research
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632.
{dyxiong, mzhang, hli}@i2r.a-star.edu.sg

Abstract

Automatic error detection is desired in the post-processing to improve machine translation quality. The previous work is largely based on confidence estimation using system-based features, such as word posterior probabilities calculated from N-best lists or word lattices. We propose to incorporate two groups of linguistic features, which convey information from outside machine translation systems, into error detection: lexical and syntactic features. We use a maximum entropy classifier to predict translation errors by integrating word posterior probability feature and linguistic features. The experimental results show that 1) linguistic features alone outperform word posterior probability based confidence estimation in error detection; and 2) linguistic features can further provide complementary information when combined with word confidence scores, which collectively reduce the classification error rate by 18.52% and improve the F measure by 16.37%.

1 Introduction

Translation hypotheses generated by a statistical machine translation (SMT) system always contain both correct parts (e.g. words, n-grams, phrases matched with reference translations) and incorrect parts. Automatically distinguishing incorrect parts from correct parts is therefore very desirable not only for post-editing and interactive machine translation (Ueffing and Ney, 2007) but also for SMT itself: either by rescoring hypotheses in the N-best list using the probability of correctness calculated for each hypothesis (Zens and Ney, 2006) or by generating new hypotheses using N-best lists from one SMT system or multiple systems (Akibay et al., 2004; Jayaraman and Lavie, 2005).

In this paper we restrict the "parts" to words. That is, we detect errors at the word level for SMT. A common approach to SMT error detection at the word level is calculating the confidence at which a word is correct. The majority of word confidence estimation methods follows three steps:

1) Calculate features that express the correctness of words either based on SMT model (e.g. translation/language model) or based on SMT system output (e.g. N-best lists, word lattices) (Blatz et al., 2003; Ueffing and Ney, 2007).

2) Combine these features together with a classification model such as multi-layer perceptron (Blatz et al., 2003), Naive Bayes (Blatz et al., 2003; Sanchis et al., 2007), or log-linear model (Ueffing and Ney, 2007).

3) Divide words into two groups (correct translations and errors) by using a classification threshold optimized on a development set.

Sometimes the step 2) is not necessary if only one effective feature is used (Ueffing and Ney, 2007); and sometimes the step 2) and 3) can be merged into a single step if we directly output predicting results from binary classifiers instead of making thresholding decision.

Various features from different SMT models and system outputs are investigated (Blatz et al., 2003; Ueffing and Ney, 2007; Sanchis et al., 2007; Raybaud et al., 2009). Experimental results show that they are useful for error detection. However, it is not adequate to just use these features as discussed in (Shi and Zhou, 2005) because the information that they carry is either from the inner components of SMT systems or from system outputs. To some extent, it has already been considered by SMT systems. Hence finding external information

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 604–611,
Uppsala, Sweden, 11-16 July 2010.

sources from outside SMT systems is desired for error detection.

Linguistic knowledge is exactly such a good choice as an external information source. It has already been proven effective in error detection for speech recognition (Shi and Zhou, 2005). However, it is not widely used in SMT error detection. The reason is probably that people have yet to find effective linguistic features that outperform non-linguistic features such as word posterior probability features (Blatz et al., 2003; Raybaud et al., 2009). In this paper, we would like to show an effective use of linguistic features in SMT error detection.

We integrate two sets of linguistic features into a maximum entropy (MaxEnt) model and develop a MaxEnt-based binary classifier to predict the category (correct or incorrect) for each word in a generated target sentence. Our experimental results show that linguistic features substantially improve error detection and even outperform word posterior probability features. Further, they can produce additional improvements when combined with word posterior probability features.

The rest of the paper is organized as follows. In Section 2, we review the previous work on word-level confidence estimation which is used for error detection. In Section 3, we introduce our linguistic features as well as the word posterior probability feature. In Section 4, we elaborate our MaxEnt-based error detection model which combine linguistic features and word posterior probability feature together. In Section 5, we describe the SMT system which we use to generate translation hypotheses. We report our experimental results in Section 6 and conclude in Section 7.

2 Related Work

In this section, we present an overview of confidence estimation (CE) for machine translation at the word level. As we are only interested in error detection, we focus on work that uses confidence estimation approaches to detect translation errors. Of course, confidence estimation is not limited to the application of error detection, it can also be used in other scenarios, such as translation prediction in an interactive environment (Grandrabur and Foster, 2003) .

In a JHU workshop, Blatz et al. (2003) investigate using neural networks and a naive Bayes classifier to combine various confidence features for confidence estimation at the word level as well as at the sentence level. The features they use for word level CE include word posterior probabilities estimated from N-best lists, features based on SMT models, semantic features extracted from WordNet as well as simple syntactic features, i.e. parentheses and quotation mark check. Among all these features, the word posterior probability is the most effective feature, which is much better than linguistic features such as semantic features, according to their final results.

Ueffing and Ney (2007) exhaustively explore various word-level confidence measures to label each word in a generated translation hypothesis as correct or incorrect. All their measures are based on word posterior probabilities, which are estimated from 1) system output, such as word lattices or N-best lists and 2) word or phrase translation table. Their experimental results show that word posterior probabilities directly estimated from phrase translation table are better than those from system output except for the Chinese-English language pair.

Sanchis et al. (2007) adopt a smoothed naive Bayes model to combine different word posterior probability based confidence features which are estimated from N-best lists, similar to (Ueffing and Ney, 2007).

Raybaud et al. (2009) study several confidence features based on mutual information between words and n-gram and backward n-gram language model for word-level and sentence-level CE. They also explore linguistic features using information from syntactic category, tense, gender and so on. Unfortunately, such linguistic features neither improve performance at the word level nor at the sentence level.

Our work departs from the previous work in two major respects.

- We exploit various linguistic features and show that they are able to produce larger improvements than widely used system-related features such as word posterior probabilities. This is in contrast to some previous work. Yet another advantage of using linguistic features is that they are system-independent, which therefore can be used across different systems.

- We treat error detection as a complete binary classification problem. Hence we di-

rectly output prediction results from our discriminatively trained classifier without optimizing a classification threshold on a distinct development set beforehand.[1] Most previous approaches make decisions based on a pretuned classification threshold τ as follows

$$class = \begin{cases} correct, & \Phi(correct, \theta) > \tau \\ incorrect, & otherwise \end{cases}$$

where Φ is a classifier or a confidence measure and θ is the parameter set of Φ. The performance of these approaches is strongly dependent on the classification threshold.

3 Features

We explore two sets of linguistic features for each word in a machine generated translation hypothesis. The first set of linguistic features are simple lexical features. The second set of linguistic features are syntactic features which are extracted from link grammar parse. To compare with the previously widely used features, we also investigate features based on word posterior probabilities.

3.1 Lexical Features

We use the following lexical features.

- wd: word itself
- pos: part-of-speech tag from a tagger trained on WSJ corpus. [2]

For each word, we look at previous n words/tags and next n words/tags. They together form a word/tag sequence pattern. The basic idea of using these features is that words in rare patterns are more likely to be incorrect than words in frequently occurring patterns. To some extent, these two features have similar function to a target language model or pos-based target language model.

3.2 Syntactic Features

High-level linguistic knowledge such as syntactic information about a word is a very natural and promising indicator to decide whether this word is syntactically correct or not. Words occurring in an ungrammatical part of a target sentence are prone to be incorrect. The challenge of using syntactic knowledge for error detection is that machine-generated hypotheses are rarely fully grammatical. They are mixed with grammatical and ungrammatical parts, which hence are not friendly to traditional parsers trained on grammatical sentences because ungrammatical parts of a machine-generated sentence could lead to a parsing failure.

To overcome this challenge, we select the *Link Grammar* (LG) parser [3] as our syntactic parser to generate syntactic features. The LG parser produces a set of labeled links which connect pairs of words with a link grammar (Sleator and Temperley, 1993).

The main reason why we choose the LG parser is that it provides a robustness feature: *null-link* scheme. The null-link scheme allows the parser to parse a sentence even when the parser can not fully interpret the entire sentence (e.g. including ungrammatical parts). When the parser fail to parse the entire sentence, it ignores one word each time until it finds linkages for remaining words. After parsing, those ignored words are not connected to any other words. We call them *null-linked* words.

Our hypothesis is that null-linked words are prone to be syntactically incorrect. We hence straightforwardly define a syntactic feature for a word w according to its links as follows

$$link(w) = \begin{cases} yes, & w\ has\ links \\ no, & otherwise \end{cases}$$

In Figure 1 we show an example of a generated translation hypothesis with its link parse. Here links are denoted with dotted lines which are annotated with link types (e.g., Jp, Op). Bracketed words, namely "," and "including", are null-linked words.

3.3 Word Posterior Probability Features

Our word posterior probability is calculated on N-best list, which is first proposed by (Ueffing et al., 2003) and widely used in (Blatz et al., 2003; Ueffing and Ney, 2007; Sanchis et al., 2007).

Given a source sentence f, let $\{e_n\}_1^N$ be the N-best list generated by an SMT system, and let e_n^i is the i-th word in e_n. The major work of calculating word posterior probabilities is to find the Levenshtein alignment (Levenshtein, 1966) between the best hypothesis e_1 and its competing hypothesis

[1] This does not mean we do not need a development set. We do validate our feature selection and other experimental settings on the development set.

[2] Available via http://www-tsujii.is.s.u-tokyo.ac.jp/~tsuruoka/postagger/

[3] Available at http://www.link.cs.cmu.edu/link/

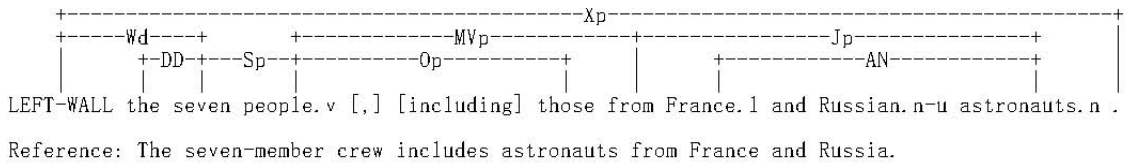


Figure 1: An example of Link Grammar parsing results.

e_n in the N-best list $\{e_n\}_1^N$. We denote the alignment between them as $\ell(e_1, e_n)$. The word in the hypothesis e_n which e_1^i is Levenshtein aligned to is denoted as $\ell_i(e_1, e_n)$.

The word posterior probability of e_1^i is then calculated by summing up the probabilities over all hypotheses containing e_1^i in a position which is Levenshtein aligned to e_1^i.

$$p_{wpp}(e_1^i) = \frac{\sum_{e_n:\, \ell_i(e_1,e_n)=e_1^i} p(e_n)}{\sum_1^N p(e_n)}$$

To use the word posterior probability in our error detection model, we need to make it discrete. We introduce a feature for a word w based on its word posterior probability as follows

$$dwpp(w) = \lfloor -log(p_{wpp}(w))/df \rfloor$$

where df is the discrete factor which can be set to 1, 0.1, 0.01 and so on. "$\lfloor\ \rfloor$" is a rounding operator which takes the largest integer that does not exceed $-log(p_{wpp}(w))/df$. We optimize the discrete factor on our development set and find the optimal value is 1. Therefore a feature "$dwpp = 2$" represents that the logarithm of the word posterior probability is between -3 and -2;

4 Error Detection with a Maximum Entropy Model

As mentioned before, we consider error detection as a binary classification task. To formalize this task, we use a feature vector ψ to represent a word w in question, and a binary variable c to indicate whether this word is correct or not. In the feature vector, we look at 2 words before and 2 words after the current word position $(w_{-2}, w_{-1}, w, w_1, w_2)$. We collect features $\{wd, pos, link, dwpp\}$ for each word among these words and combine them into the feature vector ψ for w. As such, we want the feature vector to capture the contextual environment, e.g., pos sequence pattern, syntactic pattern, where the word w occurs.

For classification, we employ the maximum entropy model (Berger et al., 1996) to predict whether a word w is correct or incorrect given its feature vector ψ.

$$p(c|\psi) = \frac{exp(\sum_i \theta_i f_i(c, \psi))}{\sum_{c'} exp(\sum_i \theta_i f_i(c', \psi))}$$

where f_i is a binary model feature defined on c and the feature vector ψ. θ_i is the weight of f_i. Table 1 shows some examples of our binary model features.

In order to learn the model feature weights θ for probability estimation, we need a training set of m samples $\{\psi^i, c^i\}_1^m$. The challenge of collecting training instances is that the correctness of a word in a generated translation hypothesis is not intuitively clear (Ueffing and Ney, 2007). We will describe the method to determine the correctness of a word in Section 6.1, which is broadly adopted in previous work.

We tune our model feature weights using an off-the-shelf MaxEnt toolkit (Zhang, 2004). To avoid overfitting, we optimize the Gaussian prior on the development set. During test, if the probability $p(correct|\psi)$ is larger than $p(incorrect|\psi)$ according the trained MaxEnt model, the word is labeled as correct otherwise incorrect.

5 SMT System

To obtain machine-generated translation hypotheses for our error detection, we use a state-of-the-art phrase-based machine translation system MOSES (Koehn et al., 2003; Koehn et al., 2007). The translation task is on the official NIST Chinese-to-English evaluation data. The training data consists of more than 4 million pairs of sentences (including 101.93M Chinese words and 112.78M English words) from LDC distributed corpora. Table 2 shows the corpora that we use for the translation task.

We build a four-gram language model using the SRILM toolkit (Stolcke, 2002), which is trained

Feature	Example
wd	$f(c,\psi) = \begin{cases} 1, & \psi.w.wd = ".", c = correct \\ 0, & otherwise \end{cases}$
pos	$f(c,\psi) = \begin{cases} 1, & \psi.w_2.pos = "NN", c = incorrect \\ 0, & otherwise \end{cases}$
$link$	$f(c,\psi) = \begin{cases} 1, & \psi.w.link = no, c = incorrect \\ 0, & otherwise \end{cases}$
$dwpp$	$f(c,\psi) = \begin{cases} 1, & \psi.w_{-2}.dwpp = 2, c = correct \\ 0, & otherwise \end{cases}$

Table 1: Examples of model features.

LDC ID	Description
LDC2004E12	United Nations
LDC2004T08	Hong Kong News
LDC2005T10	Sinorama Magazine
LDC2003E14	FBIS
LDC2002E18	Xinhua News V1 beta
LDC2005T06	Chinese News Translation
LDC2003E07	Chinese Treebank
LDC2004T07	Multiple Translation Chinese

Table 2: Training corpora for the translation task.

on Xinhua section of the English Gigaword corpus (181.1M words). For minimum error rate tuning (Och, 2003), we use NIST MT-02 as the development set for the translation task. In order to calculate word posterior probabilities, we generate 10,000 best lists for NIST MT-02/03/05 respectively. The performance, in terms of BLEU (Papineni et al., 2002) score, is shown in Table 4.

6 Experiments

We conducted our experiments at several levels. Starting with MaxEnt models with single linguistic feature or word posterior probability based feature, we incorporated additional features incrementally by combining features together. In doing so, we would like the experimental results not only to display the effectiveness of linguistic features for error detection but also to identify the additional contribution of each feature to the task.

6.1 Data Corpus

For the error detection task, we use the best translation hypotheses of NIST MT-02/05/03 generated by MOSES as our training, development, and test corpus respectively. The statistics about these corpora is shown in Table 3. Each translation hypothesis has four reference translations.

	Corpus	Sentences	Words
Training	MT-02	878	24,225
Development	MT-05	1082	31,321
Test	MT-03	919	25,619

Table 3: Corpus statistics (number of sentences and words) for the error detection task.

To obtain the linkage information, we run the LG parser on all translation hypotheses. We find that the LG parser can not fully parse 560 sentences (63.8%) in the training set (MT-02), 731 sentences (67.6%) in the development set (MT-05) and 660 sentences (71.8%) in the test set (MT-03). For these sentences, the LG parser will use the the null-link scheme to generate null-linked words.

To determine the true class of a word in a generated translation hypothesis, we follow (Blatz et al., 2003) to use the word error rate (**WER**). We tag a word as correct if it is aligned to itself in the Levenshtein alignment between the hypothesis and the nearest reference translation that has minimum edit distance to the hypothesis among four reference translations. Figure 2 shows the Levenshtein alignment between a machine-generated hypothesis and its nearest reference translation. The "Class" row shows the label of each word according to the alignment, where "c" and "i" represent *correct* and *incorrect* respectively.

There are several other metrics to tag single words in a translation hypothesis as correct or incorrect, such as **PER** where a word is tagged as correct if it occurs in one of reference translations with the same number of occurrences, **Set** which is a less strict variant of PER, ignoring the number of occurrences per word. In Figure 2, the two words "last year" in the hypothesis will be tagged as correct if we use the PER or Set metric since they do not consider the occurring positions of words. Our

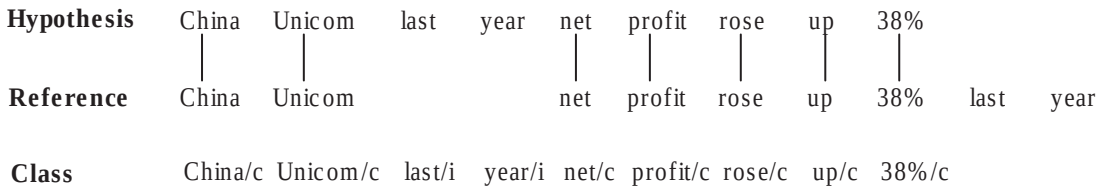


Figure 2: Tagging a word as correct/incorrect according to the Levenshtein alignment.

Corpus	BLEU (%)	RCW (%)
MT-02	33.24	47.76
MT-05	32.03	47.85
MT-03	32.86	47.57

Table 4: Case-insensitive BLEU score and ratio of correct words (RCW) on the training, development and test corpus.

metric corresponds to the **m-WER** used in (Ueffing and Ney, 2007), which is stricter than PER and Set. It is also stricter than normal WER metric which compares each hypothesis to all references, rather than the nearest reference.

Table 4 shows the case-insensitive BLEU score and the percentage of words that are labeled as correct according to the method described above on the training, development and test corpus.

6.2 Evaluation Metrics

To evaluate the overall performance of the error detection, we use the commonly used metric, classification error rate (CER) to evaluate our classifiers. CER is defined as the percentage of words that are wrongly tagged as follows

$$CER = \frac{\text{\# of wrongly tagged words}}{\text{Total \# of words}}$$

The baseline CER is determined by assuming the most frequent class for all words. Since the ratio of correct words in both the development and test set is lower than 50%, the most frequent class is "incorrect". Hence the baseline CER in our experiments is equal to the ratio of correct words as these words are wrongly tagged as incorrect.

We also use precision and recall on errors to evaluate the performance of error detection. Let n_g be the number of words of which the true class is incorrect, n_t be the number of words which are tagged as incorrect by classifiers, and n_m be the number of words tagged as incorrect that are indeed translation errors. The precision Pre is the percentage of words correctly tagged as translation errors.

$$Pre = \frac{n_m}{n_t}$$

The recall Rec is the proportion of actual translation errors that are found by classifiers.

$$Rec = \frac{n_m}{n_g}$$

F measure, the trade-off between precision and recall, is also used.

$$F = \frac{2 \times Pre \times Rec}{Pre + Rec}$$

6.3 Experimental Results

Table 5 shows the performance of our experiments on the error detection task. To compare with previous work using word posterior probabilities for confidence estimation, we carried out experiments using wpp estimated from N-best lists with the classification threshold τ, which was optimized on our development set to minimize CER. A relative improvement of 9.27% is achieved over the baseline CER, which reconfirms the effectiveness of word posterior probabilities for error detection.

We conducted three groups of experiments using the MaxEnt based error detection model with various feature combinations.

- The first group of experiments uses single feature, such as $dwpp$, pos. We find the most effective feature is pos, which achieves a 16.12% relative improvement over the baseline CER and 7.55% relative improvement over the CER of word posterior probability thresholding. Using discrete word posterior probabilities as features in the MaxEnt based error detection model is marginally better than word posterior probability thresholding in terms of CER, but obtains a 13.79% relative improvement in F measure. The syntactic feature $link$ also improves the error detection in terms of CER and particularly recall.

Combination	Features	CER (%)	Pre (%)	Rec (%)	F (%)
Baseline	-	47.57	-	-	-
Thresholding wpp	-	43.16	58.98	58.07	58.52
MaxEnt ($dwpp$)	44	43.07	56.12	81.86	66.59
MaxEnt (wd)	19,164	41.57	58.25	73.11	64.84
MaxEnt (pos)	199	39.90	58.88	79.23	67.55
MaxEnt ($link$)	19	44.31	54.72	89.72	67.98
MaxEnt ($wd + pos$)	19,363	39.43	59.36	78.60	67.64
MaxEnt ($wd + pos + link$)	19,382	39.79	58.74	80.97	68.08
MaxEnt ($dwpp + wd$)	19,208	41.04	57.18	83.75	67.96
MaxEnt ($dwpp + wd + pos$)	19,407	38.88	59.87	78.38	67.88
MaxEnt ($dwpp + wd + pos + link$)	19,426	38.76	59.89	78.94	68.10

Table 5: Performance of the error detection task.

- The second group of experiments concerns with the combination of linguistic features without word posterior probability feature. The combination of lexical features improves both CER and precision over single lexical feature (wd, pos). The addition of syntactic feature $link$ marginally undermines CER but improves recall by a lot.

- The last group of experiments concerns about the additional contribution of linguistic features to error detection with word posterior probability. We added linguistic features incrementally into the feature pool. The best performance was achieved by using all features, which has a relative of improvement of 18.52% over the baseline CER.

The first two groups of experiments show that linguistic features, individually (except for $link$) or by combination, are able to produce much better performance than word posterior probability features in both CER and F measure. The best combination of linguistic features achieves a relative improvement of 8.64% and 15.58% in CER and F measure respectively over word posterior probability thresholding.

The Table 5 also reveals how linguistic features improve error detection. The lexical features (pos, wd) improve precision when they are used. This suggests that lexical features can help the system find errors more accurately. Syntactic features ($link$), on the other hand, improve recall whenever they are used, which indicates that they can help the system find more errors.

We also show the number of features in each combination in Table 5. Except for the wd feature, the pos has the largest number of features, 199, which is a small set of features. This suggests that our error detection model can be learned from a rather small training set.

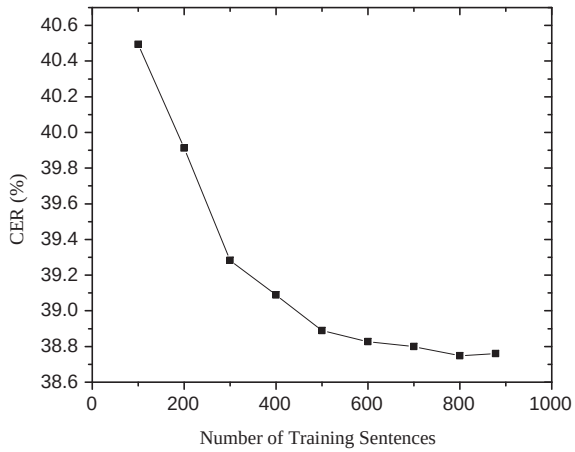


Figure 3: CER vs. the number of training sentences.

Figure 3 shows CERs for the feature combination MaxEnt ($dwpp + wd + pos + link$) when the number of training sentences is enlarged incrementally. CERs drop significantly when the number of training sentences is increased from 100 to 500. After 500 sentences are used, CERs change marginally and tend to converge.

7 Conclusions and Future Work

In this paper, we have presented a maximum entropy based approach to automatically detect errors in translation hypotheses generated by SMT

systems. We incorporate two sets of linguistic features together with word posterior probability based features into error detection.

Our experiments validate that linguistic features are very useful for error detection: 1) they by themselves achieve a higher improvement in terms of both CER and F measure than word posterior probability features; 2) the performance is further improved when they are combined with word posterior probability features.

The extracted linguistic features are quite compact, which can be learned from a small training set. Furthermore, The learned linguistic features are system-independent. Therefore our approach can be used for other machine translation systems, such as rule-based or example-based system, which generally do not produce N-best lists.

Future work in this direction involve detecting particular error types such as incorrect positions, inappropriate/unnecessary words (Elliott, 2006) and automatically correcting errors.

References

Yasuhiro Akibay, Eiichiro Sumitay, Hiromi Nakaiway, Seiichi Yamamotoy, and Hiroshi G. Okunoz. 2004. Using a Mixture of N-best Lists from Multiple MT Systems in Rank-sum-based Confidence Measure for MT Outputs. In *Proceedings of COLING.*

Adam L. Berger, Stephen A. Della Pietra andVincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics, 22(1): 39-71.*

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, Nicola Ueffing. 2003. Confidence estimation for machine translation. final report, jhu/clsp summer workshop.

Debra Elliott. 2006 Corpus-based Machine Translation Evaluation via Automated Error Detection in Output Texts. Phd Thesis, University of Leeds.

Simona Gandrabur and George Foster. 2003. Confidence Estimation for Translation Prediction. In *Proceedings of HLT-NAACL.*

S. Jayaraman and A. Lavie. 2005. Multi-engine Machine Translation Guided by Explicit Word Matching. In *Proceedings of EAMT.*

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of HLT-NAACL.*

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

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, Feb.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL 2003*.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatically Evaluation of Machine Translation. In *Proceedings of ACL 2002*.

Sylvain Raybaud, Caroline Lavecchia, David Langlois, Kamel Smaïli. 2009. Word- and Sentence-level Confidence Measures for Machine Translation. In *Proceedings of EAMT 2009.*

Alberto Sanchis, Alfons Juan and Enrique Vidal. 2007. Estimation of Confidence Measures for Machine Translation. In *Procedings of Machine Translation Summit XI.*

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. In *Proceedings of Third International Workshop on Parsing Technologies.*

Yongmei Shi and Lina Zhou. 2005. Error Detection Using Linguistic Features. In *Proceedings of HLT/EMNLP 2005.*

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of International Conference on Spoken Language Processing*, volume 2, pages 901-904.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence Measures for Statistical Machine Translation. In *Proceedings. of MT Summit IX.*

Nicola Ueffing and Hermann Ney. 2007. Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics, 33(1):9-40.*

Richard Zens and Hermann Ney. 2006. N-gram Posterior Probabilities for Statistical Machine Translation. In *HLT/NAACL: Proceedings of the Workshop on Statistical Machine Translation.*

Le Zhang. 2004. Maximum Entropy Modeling Tooklkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736 /maxent_toolkit.html.

TrustRank: Inducing Trust in Automatic Translations via Ranking

Radu Soricut
Language Weaver, Inc.
6060 Center Drive, Suite 150
Los Angeles, CA 90045
rsoricut@languageweaver.com

Abdessamad Echihabi
Language Weaver, Inc.
6060 Center Drive, Suite 150
Los Angeles, CA 90045
echihabi@languageweaver.com

Abstract

The adoption of Machine Translation technology for commercial applications is hampered by the lack of trust associated with machine-translated output. In this paper, we describe TrustRank, an MT system enhanced with a capability to rank the quality of translation outputs from good to bad. This enables the user to set a quality threshold, granting the user control over the quality of the translations.

We quantify the gains we obtain in translation quality, and show that our solution works on a wide variety of domains and language pairs.

1 Introduction

The accuracy of machine translation (MT) software has steadily increased over the last 20 years to achieve levels at which large-scale commercial applications of the technology have become feasible. However, widespread adoption of MT technology remains hampered by the lack of trust associated with machine-translated output. This lack of trust is a normal reaction to the erratic translation quality delivered by current state-of-the-art MT systems. Unfortunately, the lack of predictable quality discourages the adoption of large-scale automatic translation solutions.

Consider the case of a commercial enterprise that hosts reviews written by travellers on its web site. These reviews contain useful information about hotels, restaurants, attractions, etc. There is a large and continuous stream of reviews posted on this site, and the large majority is written in English. In addition, there is a large set of potential customers who would prefer to have these reviews available in their (non-English) native languages. As such, this enterprise presents the perfect opportunity for the deployment of a large-volume MT solution. However, travel reviews present specific challenges: the reviews tend to have poor spelling, loose grammar, and broad topics of discussion. The result is unpredictable levels of MT quality. This is undesirable for the commercial enterprise, who is not content to simply reach a broad audience, but also wants to deliver a high-quality product to that audience.

We propose the following solution. We develop TrustRank, an MT system enhanced with a capability to rank the quality of translation outputs from good to bad. This enables the user to set a quality threshold, granting the user control over the quality of the translations that it employs in its product. With this enhancement, MT adoption stops being a binary should-we-or-shouldn't-we question. Rather, each user can make a personal trade-off between the scope and the quality of their product.

2 Related Work

Work on automatic MT evaluation started with the idea of comparing automatic translations against human-produced references. Such comparisons are done either at lexical level (Papineni et al., 2002; Doddington, 2002), or at linguistically-richer levels using paraphrases (Zhou et al., 2006; Kauchak and Barzilay, 2006), WordNet (Lavie and Agarwal, 2007), or syntax (Liu and Gildea, 2005; Owczarzak et al., 2007; Yang et al., 2008; Amigó et al., 2009). In contrast, we are interested in performing MT quality assessments on documents for which reference translations are not available.

Reference-free approaches to automatic MT quality assessment, based on Machine Learning techniques such as classification (Kulesza and Shieber, 2004), regression (Albrecht and Hwa, 2007), and ranking (Ye et al., 2007; Duh, 2008), have a different focus compared to ours. Their approach, which uses a test set that is held constant and against which various MT systems are mea-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 612–621,
Uppsala, Sweden, 11-16 July 2010.

sured, focuses on evaluating system performance. Similar proposals exist outside the MT field, for instance in syntactic parsing (Ravi et al., 2008). In this case, the authors focus on estimating performance over entire test sets, which in turn is used for evaluating system performance. In contrast, we focus on evaluating the quality of the translations themselves, while the MT system is kept constant.

A considerable amount of work has been done in the related area of confidence estimation for MT, for which Blatz et al. (2004) provide a good overview. The goal of this work is to identify small units of translated material (words and phrases) for which one can be confident in the quality of the translation. Related to this goal, and closest to our proposal, is the work of Gamon et al. (2005) and Specia et al. (2009). They describe Machine Learning approaches (classification and regression, respectively) aimed at predicting which sentences are likely to be well/poorly translated. Our work, however, departs from all these works in several important aspects.

First, we want to make the quality predictions at document-level, as opposed to sentence-level (Gamon et al., 2005; Specia et al., 2009), or word/phrase-level (Blatz et al., 2004; Ueffing and Ney, 2005). Document-level granularity is a requirement for large-scale commercial applications that use fully-automated translation solutions. For these applications, the need to make the distinction between "good translation" and "poor translation" must be done at document level. Otherwise, it is not actionable. In contrast, quality-prediction or confidence estimation at sentence- or word-level fits best a scenario in which automated translation is only a part of a larger pipeline. Such pipelines usually involve human post-editing, and are useful for translation productivity (Lagarda et al., 2009). Such solutions, however, suffer from the inherent volume bottleneck associated with human involvement. Our fully-automated solution targets large volume translation needs, on the order of 10,000 documents/day or more.

Second, we use automatically generated training labels for the supervised Machine Learning approach. In the experiments presented in this paper, we use BLEU scores (Papineni et al., 2002) as training labels. However, they can be substituted with any of the proposed MT metrics that use human-produced references to automatically assess translation quality (Doddington, 2002; Lavie and Agarwal, 2007). In a similar manner, the work of (Specia et al., 2009) uses NIST scores, and the work of (Ravi et al., 2008) uses PARSEVAL scores. The main advantage of this approach is that we can generate quickly and cheaply as many learning examples as needed. Additionally, we can customize the prediction models on a large variety of genres and domains, and quickly scale to multiple language pairs. In contrast, solutions that require training labels produced manually by humans (Gamon et al., 2005; Albrecht and Hwa, 2007) have difficulties producing prediction models fast enough, trained on enough data, and customized for specific domains.

Third, the main metric we use to assess the performance of our solution is targeted directly at measuring translation quality gains. We are interested in the extrinsic evaluation of the quantitative impact of the TrustRank solution, rather than in the intrinsic evaluation of prediction errors (Ravi et al., 2008; Specia et al., 2009).

3 Experimental Framework

3.1 Domains

We are interested in measuring the impact of TrustRank on a variety of genres, domains, and language pairs. Therefore, we set up the experimental framework accordingly. We use three proprietary data sets, taken from the domains of Travel (consumer reviews), Consumer Electronics (customer support for computers, data storage, printers, etc.), and HighTech (customer support for high-tech components). All these data sets come in a variety of European and Asian language pairs. We also use the publicly available data set used in the WMT09 task (Koehn and Haddow, 2009) (a combination of European parliament and news data). Information regarding the sizes of these data sets is provided in Table 2.

3.2 Metrics

We first present the experimental framework designed to answer the main question we want to address: can we automatically produce a ranking for document translations (for which no human-produced references are available), such that the translation quality of the documents at the top of this ranking is higher than the average translation quality? To this end, we use several metrics that can gauge how well we answer this question.

The first metric is Ranking Accuracy (rAcc), see (Gunawardana and Shani, 2009). We are interested in ranking N documents and assigning them into n quantiles. The formula is:

$$\text{rAcc}[n] = \text{Avg}_{i=1}^{n} \frac{\text{TP}_i}{\frac{N}{n}} = \frac{1}{N} \times \Sigma_{i=1}^{n} \text{TP}_i$$

where TP_i (True-Positive$_i$) is the number of correctly-assigned documents in quantile i. Intuitively, this formula is an average of the ratio of documents correctly assigned in each quantile.

The rAcc metric provides easy to understand lowerbounds and upperbounds. For example, with a method that assigns random ranks, when using 4 quantiles, the accuracy is 25% in any of the quantiles, hence an rAcc of 25%. With an oracle-based ranking, the accuracy is 100% in any of the quantiles, hence an rAcc of 100%. Therefore, the performance of any decent ranking method, when using 4 quantiles, can be expected to fall somewhere between these bounds.

The second and main metric is the volume-weighted BLEU gain (vBLEUΔ) metric. It measures the average BLEU gain when trading-off volume for accuracy on a predefined scale. The general formula, for n quantiles, is

$$\text{vBLEU}\Delta[n] = \Sigma_{i=1}^{n-1} w_i \times (\text{BLEU}_{1...i} - \text{BLEU})$$

$$\text{with } w_i = \frac{\frac{i}{n}}{\Sigma_{j=1}^{n-1} \frac{j}{n}} = \frac{i}{\Sigma_{j=1}^{n-1} j} = \frac{2i}{n(n-1)}$$

where $\text{BLEU}_{1...i}$ is the BLEU score of the first i quantiles, and BLEU is the score over all the quantiles. Intuitively, this formula provides a volume-weighted average of the BLEU gain obtained while varying the threshold of acceptance from 1 to n-1. (A threshold of acceptance set to the n-th quantile means accepting all the translations and therefore ignore the rankings, so we do not include it in the average.) Without rankings (or with random ranks), the expected vBLEU$\Delta[n]$ is zero, as the value $\text{BLEU}_{1...i}$ is expected to be the same as the overall BLEU for any i. With oracle ranking, the expected vBLEU$\Delta[n]$ is a positive number representative of the upperbound on the quality of the translations that pass an acceptance threshold. We report the vBLEU$\Delta[n]$ values as signed numbers, both within a domain and when computed as an average across domains.

The choice regarding the number of quantiles is closely related to the choice of setting an acceptance quality threshold. Because we want the solution to stay unchanged while the acceptance quality threshold can vary, we cannot treat this as a classification problem. Instead, we need to provide a complete ranking over an input set of documents. As already mentioned, TrustRank uses a regression method that is trained on BLEU scores as training labels. The regression functions are then used to predict a BLEU-like number for each document in the input set. The rankings are derived trivially from the predicted BLEU numbers, by simply sorting from highest to lowest. Reference ranking is obtained similarly, using actual BLEU scores.

Although we are mainly interested in the ranking problem here, it helps to look at the error produced by the regression models to arrive at a more complete picture. Besides the two metrics for ranking described above, we use the well-known regression metrics MAE (mean absolute error) and TE (test-level error):

$$\text{MAE} = \frac{1}{N} \times \Sigma_{k=1}^{N} |\text{predBLEU}_k - \text{BLEU}_k|$$

$$\text{TE} = \text{predBLEU} - \text{BLEU}$$

where BLEU_k is the BLEU score for document k, predBLEU_k is the predicted BLEU value, and predBLEU is a weighted average of the predicted document-level BLEU numbers over the entire set of N documents.

3.3 Experimental conditions

The MT system used by TrustRank (TrustRank-MT) is a statistical phrase-based MT system similar to (Och and Ney, 2004). As a reference point regarding the performance of this system, we use the official WMT09 parallel data, monolingual data, and development tuning set (news-dev2009a) to train baseline TrustRank-MT systems for each of the ten WMT09 language pairs. Our system produces translations that are competitive with state-of-the-art systems. We show our baseline-system BLEU scores on the official development test set (news-dev2009b) for the WMT09 task in Table 1, along with the BLEU scores reported for the baseline Moses system (Koehn and Haddow, 2009).

For each of the domains we consider, we partition the data sets as follows. We first set aside 3000 documents, which we call the Regression set [1]. The remaining data is called the training MT

[1]For parallel data for which we do not have document

From Eng	Fra	Spa	Ger	Cze	Hun
Moses	17.8	22.4	13.5	11.4	6.5
TrustRank-MT	21.3	22.8	14.3	9.1	8.5
Into Eng	Fra	Spa	Ger	Cze	Hun
Moses	21.2	22.5	16.6	16.9	8.8
TrustRank-MT	22.4	23.8	19.8	13.3	10.4

Table 1: BLEU scores (uncased) for the TrustRank-MT system compared to Moses (WMT09 data).

set, on which the MT system is trained. From the Regression set, we set aside 1000 parallel documents to be used as a blind test set (called Regression Test) for our experiments. An additional set of 1000 parallel documents is used as a development set, and the rest of 1000 parallel documents is used as the regression-model training set.

We have also performed learning-curve experiments using between 100 and 2000 documents for regression-model training. We do not go into the details of these experiments here for lack of space. The conclusion derived from these experiments is that 1000 documents is the point where the learning-curves level off.

In Table 2, we provide a few data points with respect to the data size of these sets (tokenized word-count on the source side). We also report the BLEU performance of the TrustRank-MT system on the Regression Test set.

Note that the differences between the BLEU scores reported in Table 1 and the BLEU scores under the WMT09 label in Table 2 reflect differences in the genres of these sets. The official development test set (news-dev2009b) for the WMT09 task is news only. The regression Test sets have the same distribution between Europarl data and news as the corresponding training data set for each language pair.

4 The ranking algorithm

As mentioned before, TrustRank takes a supervised Machine Learning approach. We automatically generate the training labels by computing BLEU scores for every document in the Regression training set.

boundaries, we simply simulate document boundaries after every 10 consecutive sentences.

LP	MT set	Regression set		
	Train	Train	Test	BLEU
WMT09				
Eng-Spa	41Mw	277Kw	281Kw	41.0
Eng-Fra	41Mw	282Kw	283Kw	37.1
Eng-Ger	41Mw	282Kw	280Kw	23.7
Eng-Cze	1.2Mw	241Kw	242Kw	10.3
Eng-Hun	30Mw	209Kw	206Kw	14.5
Spa-Eng	42Mw	287Kw	293Kw	40.1
Fra-Eng	44Mw	305Kw	308Kw	37.9
Ger-Eng	39Mw	269Kw	267Kw	29.4
Cze-Eng	1.0Mw	218Kw	219Kw	19.7
Hun-Eng	26Mw	177Kw	176Kw	24.0
Travel				
Eng-Spa	4.3Mw	123Kw	121Kw	31.2
Eng-Fra	3.5Mw	132Kw	126Kw	27.8
Eng-Ita	3.4Mw	179Kw	183Kw	22.5
Eng-Por	13.1Mw	83Kw	83Kw	41.9
Eng-Ger	7.0Mw	69Kw	69Kw	27.6
Eng-Dut	0.7Mw	89Kw	84Kw	41.9
Electronics				
Eng-Spa	7.0Mw	150Kw	149Kw	65.2
Eng-Fra	6.5Mw	129Kw	129Kw	55.8
Eng-Ger	5.9Mw	139Kw	140Kw	42.1
Eng-Chi	7.1Mw	135Kw	136Kw	63.9
Eng-Por	2.0Mw	124Kw	115Kw	47.9
HiTech				
Eng-Spa	2.8Mw	143Kw	148Kw	59.0
Eng-Ger	5.1Mw	162Kw	155Kw	36.6
Eng-Chi	5.6Mw	131Kw	129Kw	60.6
Eng-Rus	2.8Mw	122Kw	117Kw	39.2
Eng-Kor	4.2Mw	129Kw	140Kw	49.4

Table 2: Data sizes and BLEU on Regression Test.

4.1 The learning method

The results we report here are obtained using the freely-available Weka engine [2]. We have compared and contrasted results using all the regression packages offered by Weka, including regression functions based on simple and multiple-feature Linear regression, Pace regression, RBF networks, Isotonic regression, Gaussian Processes, Support Vector Machines (with SMO optimization) with polynomial and RBF kernels, and regression trees such as REP trees and M5P trees. Due to lack of space and the tangential impact on the message of this paper, we do not report

[2] Weka software at http://www.cs.waikato.ac.nz/ml/weka/, version 3.6.1, June 2009.

these contrastive experiments here.

The learning technique that consistently yields the best results is M5P regression trees (weka.classifiers.trees.M5P). Therefore, we report all the results in this paper using this learning method. As an additional advantage, the decision trees and the regression models produced in training are easy to read, understand, and interpret. One can get a good insight into what the impact of a certain feature on a final predicted value is by simply inspecting these trees.

4.2 The features

In contrast to most of the work on confidence estimation (Blatz et al., 2004), the features we use are not internal features of the MT system. Therefore, TrustRank can be applied for a large variety of MT approaches, from statistical-based to rule-based approaches.

The features we use can be divided into text-based, language-model–based, pseudo-reference–based, example-based, and training-data–based feature types. These feature types can be computed either on the source-side (input documents) or on the target-side (translated documents).

Text-based features

These features simply look at the length of the input in terms of (tokenized) number of words. They can be applied on the input, where they induce a correlation between the number of words in the input document and the expected BLEU score for that document size. They can also be applied on the produced output, and learn a similar correlation for the produced translation.

Language-model–based features

These features are among the ones that were first proposed as possible differentiators between good and bad translations (Gamon et al., 2005). They are a measure of how likely a collection of strings is under a language model trained on monolingual data (either on the source or target side).

The language-model–based feature values we use here are computed as document-level perplexity numbers using a 5-gram language model trained on the MT training set.

Pseudo-reference–based features

Previous work has shown that, in the absence of human-produced references, automatically-produced ones are still helpful in differentiating between good and bad translations (Albrecht and Hwa, 2008). When computed on the target side, this type of features requires one or more secondary MT systems, used to generate translations starting from the same input. These pseudo-references are useful in gauging translation convergence, using BLEU scores as feature values. In intuitive terms, their usefulness can be summarized as follows: "if system X produced a translation A and system Y produced a translation B starting from the same input, and A and B are similar, then A is probably a good translation".

An important property here is that systems X and Y need to be as different as possible from each other. This property ensures that a convergence on similar translations is not just an artifact, but a true indication that the translations are correct. The secondary systems we use here are still phrase-based, but equipped with linguistically-oriented modules similar with the ones proposed in (Collins et al., 2005; Xu et al., 2009).

The source-side pseudo-reference–based feature type is of a slightly different nature. It still requires one or more secondary MT systems, but operating in the reverse direction. A translated document produced by the main MT system is fed to the secondary MT system(s), translated back into the original source language, and used as pseudo-reference(s) when computing a BLEU score for the original input. In intuitive terms: "if system X takes document A and produces B, and system X^{-1} takes B and produces C, and A and C are similar, then B is probably a good translation".

Example-based features

For example-based features, we use a development set of 1000 parallel documents, for which we produce translations and compute document-level BLEU scores. We set aside the top-100 BLEU scoring documents and bottom-100 BLEU scoring documents. They are used as positive examples (with better-than-average BLEU) and negative examples (with worse-than-average BLEU), respectively. We define a positive-example–based feature function as a geometric mean of 1-to-4–gram precision scores (i.e., BLEU score without length penalty) between a document (on either source or target side) and the positive examples used as references (similarly for negative-example–based features).

The intuition behind these features can be summarized as follows: "if system X translated docu-

ment A well/poorly, and A and B are similar, then system X probably translates B well/poorly".

Training-data–based features

If the main MT system is trained on a parallel corpus, the data in this corpus can be exploited towards assessing translation quality (Specia et al., 2009). In our context, the documents that make up this corpus can be used in a fashion similar with the positive examples. One type of training-data–based features operates by computing the number of out-of-vocabulary (OOV) tokens with respect to the training data (on either source or target side).

A more powerful type of training-data–based features operates by computing a BLEU score between a document (source or target side) and the training-data documents used as references. Intuitively, we assess the coverage with respect to the training data and correlate it with a BLEU score: "if the n-grams of input document A are well covered by the source-side of the training data, the translation of A is probably good" (on the source side); "if the n-grams in the output translation B are well covered by the target-side of the parallel training data, then B is probably a good translation" (on the target side).

4.3 Results

We are interested in the best performance for TrustRank using the features described above. In this section, we focus on reporting the results obtain for the English-Spanish language pair. In the next section, we report results obtained on all the language pairs we considered.

Before we discuss the results of TrustRank, let us anchor the numerical values using some lower- and upper-bounds. As a baseline, we use a regression function that outputs a constant number for each document, equal to the BLEU score of the Regression Training set. As an upperbound, we use an oracle regression function that outputs a number for each document that is equal to the actual BLEU score of that document. In Table 4, we present the performance of these regression functions across all the domains considered.

As already mentioned, the rAcc values are bounded by the 25% lowerbound and the 100% upperbound. The vBLEUΔ values are bounded by 0 as lowerbound, and some positive BLEU gain value that varies among the domains we considered from +6.4 (Travel) to +13.5 (HiTech).

The best performance obtained by TrustRank for English-Spanish, using all the features described, is presented in Table 3. The ranking accuracy numbers on a per-quantile basis reveals an important property for the approach we advocate. The ranking accuracy on the first quantile Q_1 (identifying the best 25% of the translations) is 52% on average across the domains. For the last quantile Q_4 (identifying the worst 25% of the translations), it is 56%. This is much better than the ranking accuracy for the median-quality translations (35-37% accuracy for the two middle quantiles). This property fits well our scenario, in which we are interested in associating trust in the quality of the translations in the top quantile.

Domain	rAcc	vBLEUΔ[4]	MAE	TE
Baseline				
WMT09	25%	0	9.9	+0.4
Travel	25%	0	8.3	+2.0
Electr.	25%	0	12.2	+2.6
HiTech	25%	0	16.9	+2.4
Dom. avg.	25%	0	11.8	1.9
Oracle				
WMT09	100%	+8.2	0	0
Travel	100%	+6.4	0	0
Electr.	100%	+9.2	0	0
HiTech	100%	+13.5	0	0
Dom. avg.	100%	+9.3	0	0

Table 4: Lower- and upper-bounds for ranking and regression accuracy (English-Spanish).

The quality of the top quantile translations is quantifiable in terms of BLEU gain. The 250 document translations in Q_1 for Travel have a BLEU score of 38.0, a +6.8 BLEU gain compared to the overall BLEU of 31.2 (Q_{1-4}). The Q_1 HiTech translations, with a BLEU of 77.9, have a +18.9 BLEU gain compared to the overall BLEU of 59.0. The TrustRank algorithm allows us to tradeoff quantity versus quality on any scale. The results under the BLEU heading in Table 3 represent an instantiation of this ability to a 3-point scale (Q_1,Q_{1-2},Q_{1-3}). The vBLEUΔ numbers reflect an average of the BLEU gains for this instantiation (e.g., a +11.6 volume-weighted average BLEU gain for the HiTech domain).

We are also interested in the best performance under more restricted conditions, such as time constraints. The assumption we make here is that the translation time dwarfs the time needed for fea-

Domain	Ranking Accuracy					Translation Accuracy					MAE	TE
						BLEU				vBLEUΔ[4]		
	Q_1	Q_2	Q_3	Q_4	rAcc	Q_1	Q_{1-2}	Q_{1-3}	Q_{1-4}			
WMT09	34%	26%	29%	40%	32%	44.8	43.6	42.4	41.1	+2.1	9.6	-0.1
Travel	50%	26%	29%	41%	36%	38.0	35.1	33.0	31.2	+3.4	7.4	-1.9
Electronics	57%	38%	39%	68%	51%	76.1	72.7	69.6	65.2	+6.5	8.4	-2.6
HiTech	65%	48%	49%	75%	59%	77.9	72.7	66.7	59.0	+11.6	8.6	-2.1
Dom. avg.	**52%**	**35%**	**37%**	**56%**	**45%**	-				**+5.9**	**8.5**	**1.7**

Table 3: Detailed performance using all features (English-Spanish).

ture and regression value computation. Therefore, the most time-expensive feature is the source-side pseudo-reference–based feature, which effectively doubles the translation time required. Under the "time-constrained" condition, we exclude this feature and use all of the remaining features. Table 5 presents the results obtained for English-Spanish.

Domain	rAcc	vBLEUΔ[4]	MAE	TE
"Time-constrained" condition				
WMT09	32%	+2.1	9.6	-0.1
Travel	35%	+3.2	7.4	-1.8
Electronics	50%	+6.3	8.4	-2.2
HiTech	59%	+11.6	8.9	-2.1
Dom. avg.	**44%**	**+5.8**	**8.6**	**1.6**

Table 5: "Time-constrained" performance (English-Spanish).

The results presented above allow us to draw a series of conclusions.

Benefits vary by domain

Even with oracle rankings (Table 4), the benefits vary from one domain to the next. For Travel, with an overall BLEU score in the low 30s (31.2), we stand to gain at most +6.4 BLEU points on average (+6.4 vBLEUΔ upperbound). For a domain such as HiTech, even with a high overall BLEU score close to 60 (59.0), we stand to gain twice as much (+13.5 vBLEUΔ upperbound).

Performance varies by domain

As the results in Table 3 show, the best performance we obtain also varies from one domain to the next. For instance, the ranking accuracy for the WMT09 domain is only 32%, while for the HiTech domain is 59%. Also, the BLEU gain for the WMT09 domain is only +2.1 vBLEUΔ (compared to the upperbound vBLEUΔ of +8.2, it is only 26% of the oracle performance). In contrast, the BLEU gain for the HiTech domain is +11.6 vBLEUΔ (compared to the +13.5 vBLEUΔ upperbound, it is 86% of the oracle performance).

Positive feature synergy and overlap

The features we described capture different information, and their combination achieves the best performance. For instance, in the Electronics domain, the best single feature is the target-side n-gram coverage feature, with +5.3 vBLEUΔ. The combination of all features gives a +6.5 vBLEUΔ.

The numbers in Table 3 also show that eliminating some of the features results in lower performance. The rAcc drops from 45% to 44% in under the "time-constraint" condition (Table 5). The difference in the rankings is statistically significant at $p < 0.01$ using the Wilcoxon test (Demšar, 2006).

However, this drop is quantitatively small (1% rAcc drop, -0.1 in vBLEUΔ, averaged across domains). This suggests that, even when eliminating features that by themselves have a good discriminatory power (the source-side pseudo-reference–based feature achieves a +5.0 vBLEUΔ as a single feature in the Electronics domain), the other features compensate to a large degree.

Poor regression performance

By looking at the results of the regression metrics, we conclude that the predicted BLEU numbers are not accurate in absolute value. The aggregated Mean Absolute Error (MAE) is 8.5 when using all the features. This is less than the baseline MAE of 11.8, but it is too high to allow us to confidently use the document-level BLEU numbers as reliable indicators of translation accuracy. The Test Error (TE) numbers are not encouraging either, as the 1.7 TE of TrustRank is close to the baseline TE of 1.9 (see Table 4 for baseline numbers).

5 Large-scale experimental results

In this section, we present the performance of TrustRank on a variety of language pairs (Table 6). We report the BLEU score obtained on our 1000-document regression Test, as well as ranking and regression performance using the rAcc, vBLEUΔ, MAE, and TE metrics.

As the numbers for the ranking and regression metrics show, the same trends we observed for English-Spanish hold for many other language pairs as well. Some domains, such as HiTech, are easier to rank regardless of the language pair, and the quality gains are consistently high (+9.9 average vBLEUΔ for the 5 language pairs considered). Other domains, such as WMT09 and Travel, are more difficult to rank. However, the WMT09 English-Hungarian data set appears to be better suited for ranking, as the vBLEUΔ numbers are higher compared to the rest of the language pairs from this domain (+4.3 vBLEUΔ for Eng-Hun, +7.1 vBLEUΔ for Hun-Eng). For Travel, English-Dutch is also an outlier in terms of quality gains (+12.9 vBLEUΔ).

Overall, the results indicate that TrustRank obtains consistent performance across a large variety of language pairs. Similar with the conclusion for English-Spanish, the regression performance is currently too poor to allow us to confidently use the absolute document-level predicted BLEU numbers as indicators of translation accuracy.

6 Examples and Illustrations

As the experimental results in Table 6 show, the regression performance varies considerably across domains. Even within the same domain, the nature of the material used to perform the experiments can influence considerably the results we obtain. In Figure 1, we plot ⟨BLEU,predBLEU⟩ points for three of our language pairs presented in Table 6: Travel Eng-Fra, Travel Eng-Dut, and HiTech Eng-Rus. These plots illustrate the tendency of the predicted BLEU values to correlate with the actual BLEU scores. The amount of correlation visible in these plots matches the performance numbers provided in Table 6, with Travel Eng-Fra at a lower level of correlation compared to Travel Eng-Dut and HiTech Eng-Rus. The ⟨BLEU,predBLEU⟩ points tend to align along a line at an angle smaller than 45°, an indication of the fact that the BLEU predictions tend to be more conservative compared to the actual BLEU scores. For example, in the Travel Eng-Fra case, the predicted BLEU numbers are spread across a narrower band (95% of the values are in the [19-35] interval), compared to the actual BLEU scores (95% of the values are in the [11-47] interval).

Domain	BLEU	rAcc	vBLEUΔ[4]	MAE	TE
WMT09					
Eng-Spa	41.0	35%	+2.4	9.2	-0.3
Eng-Fra	37.1	37%	+3.3	8.3	-0.5
Eng-Ger	23.7	32%	+1.9	5.8	-0.7
Eng-Cze	10.3	38%	+1.3	3.1	-0.6
Eng-Hun	14.5	55%	+4.3	3.7	-1.1
Spa-Eng	40.1	37%	+3.3	8.1	-0.2
Fra-Eng	37.9	39%	+3.8	10.1	-0.6
Ger-Eng	29.4	36%	+2.7	5.9	-0.9
Cze-Eng	19.7	40%	+2.4	4.3	-0.6
Hun-Eng	24.0	61%	+7.1	4.9	-1.8
Travel					
Eng-Spa	31.2	36%	+3.4	7.4	-1.9
Eng-Fra	27.8	39%	+2.7	6.2	-0.9
Eng-Ita	22.5	39%	+2.4	5.1	+0.0
Eng-Por	41.9	51%	+5.6	8.6	+1.1
Eng-Ger	27.6	37%	+5.7	11.8	-0.4
Eng-Dut	41.9	52%	+12.9	12.9	-0.7
Electronics					
Eng-Spa	65.2	51%	+6.5	8.4	-2.6
Eng-Fra	55.8	49%	+7.7	8.4	-2.3
Eng-Ger	42.1	57%	+8.9	7.4	-1.6
Eng-Chi	63.9	48%	+6.4	8.6	-0.8
Eng-Por	47.9	49%	+6.9	9.0	-1.8
HiTech					
Eng-Spa	59.0	59%	+11.6	8.6	-2.1
Eng-Ger	36.6	62%	+9.2	7.1	-1.0
Eng-Chi	60.3	54%	+7.5	8.4	-1.0
Eng-Rus	39.2	62%	+10.7	8.7	-2.1
Eng-Kor	49.4	61%	+10.5	9.7	-3.2

Table 6: Performance of TrustRank on a variety of domains and language pairs.

These intervals are also useful for gauging the level of difficulty stemming from the nature of the material used to perform the experiments. In the case of Travel Eng-Fra, the actual BLEU scores are clustered in a narrower band (interval [11-47] covers 95% of the values), compared to the actual BLEU scores for Travel Eng-Dut (interval [11-92] covers 95% of the values) and HiTech Eng-Rus (interval [3-80] covers 95% of the values). This

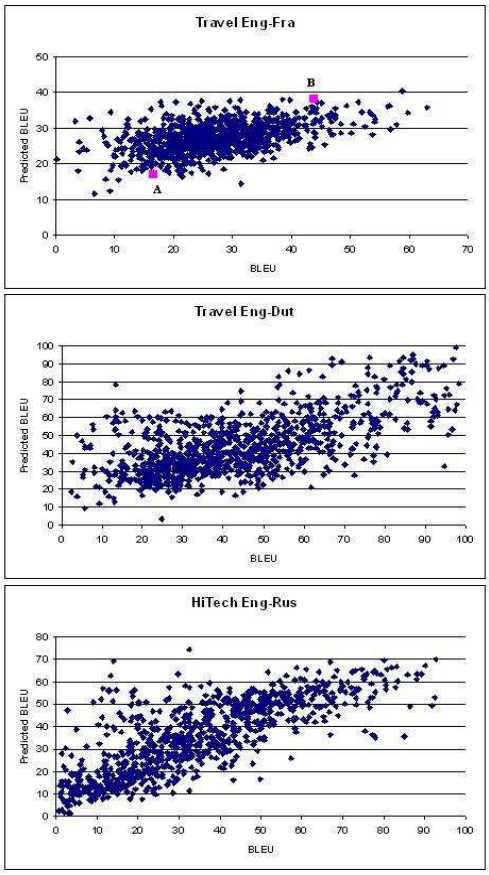


Figure 1: Examples of BLEU versus predBLEU.

means that the documents in the latter cases are easier to distinguish, compared to the documents in Travel Eng-Fra.

To provide an intuitive feel for the difference between the level of translation performance between documents ranked close to the bottom and documents ranked close to the top, we present here two example translations. They are documents that we randomly picked from the bottom 10% and top 10% of the Travel Eng-Fra document set, and they correspond to points A and B in the first plot of Figure 1, respectively. The A-Fra and B-Fra entries below are produced by our Eng-Fra TrustRank-MT system, starting from A-Eng and B-Eng[3], respectively.

A-Eng This will be our 18th year,still love it. Same hotel, room, staff, even other guests from other countries, its lovely to see everyone that you have gotten to know over the years, even if ,you or they ,do not speak each others language. We love the Island some much that, hopefuly, that is where we are retiring to, we do keep looking for that affordable place.

A-Fra Ce sera notre 18ème année, adore. Même hôtel, la chambre, le personnel, même d'autres clients dans d'autres pays, c'est très agréable de voir que tout le monde vous aurais savoir au cours de ces dernières années, même si, ou bien ils vous, ne parlent pas chaque d'autres langues. Nous adorons l'île des que, hopefuly, c'est l'endroit où nous avons retiring, nous ne pour chercher un endroit abordable.

B-Eng Stayed at the Intercontinental for 4 nights. It is in an excellent location, not far from the French Quarter. The rooms are large, clean, and comfortable. The staff is friendly and helpful. Parking is very expensive, around $29. 00 a day. There is a garage next door which is a little more reasonable. I certainly suggest this hotel to others.

B-Fra J'ai séjourné à l'Intercontinental pour 4 nuits. Il est très bien situé, pas loin du Quartier Français. Les chambres sont grandes, propres et confortables. Le personnel est sympa et serviable. Le parking est très cher, autour de 29 $ par jour. Il y a un garage à côté, ce qui est un peu plus raisonnable. Je conseille cet hôtel à d'autres.

Document A-Fra is a poor translation, and is ranked in the bottom 10%, while document B-Fra is a nearly-perfect translation ranked in the top 10%, out of a total of 1000 documents.

7 Conclusions and Future Work

Commercial adoption of MT technology requires trust in the translation quality. Rather than delay this adoption until MT attains a near-human level of sophistication, we propose an interim approach. We present a mechanism that allows MT users to trade quantity for quality, using automatically-determined translation quality rankings.

The results we present in this paper show that document-level translation quality rankings provide quantitatively strong gains in translation quality, as measured by BLEU. A difference of +18.9 BLEU, like the one we obtain for the English-Spanish HiTech domain (Table 3), is persuasive evidence for inspiring trust in the quality of selected translations. This approach enables us to develop TrustRank, a complete MT solution that enhances automatic translation with the ability to identify document subsets containing translations that pass an acceptable quality threshold.

When measuring the performance of our solution across several domains, it becomes clear that some domains allow for more accurate quality prediction than others. Given the immediate benefit that can be derived from increasing the ranking accuracy for translation quality, we plan to open up publicly available benchmark data that can be used to stimulate and rigorously monitor progress in this direction.

[3]We preserved the original writing style of the documents in the source language.

References

Joshua Albrecht and Rebecca Hwa. 2007. Regression for sentence-level MT evaluation with pseudo references. In *Proceedings of ACL*.

Joshua Albrecht and Rebecca Hwa. 2008. The role of pseudo references in MT evaluation. In *Proceedings of ACL*.

Enrique Amigó, Jesús Giménez, Julio Gonzalo, and Felisa Verdejo. 2009. The contribution of linguistic features to automatic machine translation evaluation. In *Proceedings of ACL*.

John Blatz, Erin Fitzgerald, GEorge Foster, Simona Gandrabur, Cyril Gouette, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of COLING*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

J. Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram coocurrence statistics. In *Proceedings of HLT*.

Kevin Duh. 2008. Ranking vs. regression in machine translation evaluation. In *Proceedings of the ACL Third Workshop on Statistical Machine Translation*.

Michael Gamon, Anthony Aue, and Martine Smets. 2005. Sentence-level MT evaluation without reference translations: Beyond language modeling. In *Proceedings of EAMT*.

Asela Gunawardana and Guy Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of HLT/NAACL*.

Philipp Koehn and Barry Haddow. 2009. Edinburgh's submission to all tracks of the WMT2009 shared task with reordering and speed improvements to Moses. In *Proceedings of EACL Workshop on Statistical Machine Translation*.

Alex Kulesza and Stuart M. Shieber. 2004. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*.

A.-L. Lagarda, V. Alabau, F. Casacuberta, R. Silva, and E. Díaz de Liaño. 2009. Statistical post-editing of a rule-based machine translation system. In *Proceedings of HLT/NAACL*.

A. Lavie and A. Agarwal. 2007. METEOR: An autoamtic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of ACL Workshop on Statistical Machine Translation*.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translations. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Franz Joseph Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Karolina Owczarzak, Josef Genabith, and Andy Way. 2007. Evaluating machine translation with LFG dependencies. *Machine Translation*, 21(2):95–119, June.

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

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic prediction of parsing accuracy. In *Proceedings of EMNLP*.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marcho Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation. In *Proceedings of EAMT*.

Nicola Ueffing and Hermann Ney. 2005. Application of word-level confidence measures in interactive statistical machine translation. In *Proceedings of EAMT*.

Peng Xu, Jaeho Kang, Michael Ringaard, and Franz Och. 2009. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proceedings of ACL*.

Muyun Yang, Shuqi Sun, Jufeng Li, Sheng Li, and Zhao Tiejun. 2008. A linguistically motivated MT evaluation system based on SVM regression. In *Proceedings of AMTA*.

Yang Ye, Ming Zhou, and Chin-Yew Lin. 2007. Sentence level machine translation evaluation as a ranking. In *Proceedings of the ACL Second Workshop on Statistical Machine Translation*.

Liang Zhou, Chin-Yew Lin, and Eduard Hovy. 2006. Re-evaluating machine translation results with paraphrase support. In *Proceedings of EMNLP*.

Bridging SMT and TM with Translation Recommendation

Yifan He **Yanjun Ma** **Josef van Genabith** **Andy Way**
Centre for Next Generation Localisation
School of Computing
Dublin City University
{yhe,yma,josef,away}@computing.dcu.ie

Abstract

We propose a translation recommendation framework to integrate Statistical Machine Translation (SMT) output with Translation Memory (TM) systems. The framework recommends SMT outputs to a TM user when it predicts that SMT outputs are more suitable for post-editing than the hits provided by the TM. We describe an implementation of this framework using an SVM binary classifier. We exploit methods to fine-tune the classifier and investigate a variety of features of different types. We rely on automatic MT evaluation metrics to approximate human judgements in our experiments. Experimental results show that our system can achieve 0.85 precision at 0.89 recall, excluding exact matches. Furthermore, it is possible for the end-user to achieve a desired balance between precision and recall by adjusting confidence levels.

1 Introduction

Recent years have witnessed rapid developments in statistical machine translation (SMT), with considerable improvements in translation quality. For certain language pairs and applications, automated translations are now beginning to be considered acceptable, especially in domains where abundant parallel corpora exist.

However, these advances are being adopted only slowly and somewhat reluctantly in professional localization and post-editing environments. Post-editors have long relied on translation memories (TMs) as the main technology assisting translation, and are understandably reluctant to give them up. There are several simple reasons for this: 1) TMs are useful; 2) TMs represent considerable effort and investment by a company or (even more so) an individual translator; 3) the fuzzy match score used in TMs offers a good approximation of post-editing effort, which is useful both for translators and translation cost estimation and, 4) current SMT translation confidence estimation measures are not as robust as TM fuzzy match scores and professional translators are thus not ready to replace fuzzy match scores with SMT internal quality measures.

There has been some research to address this issue, see e.g. (Specia et al., 2009a) and (Specia et al., 2009b). However, to date most of the research has focused on better confidence measures for MT, e.g. based on training regression models to perform confidence estimation on scores assigned by post-editors (cf. Section 2).

In this paper, we try to address the problem from a different perspective. Given that most post-editing work is (still) based on TM output, we propose to recommend MT outputs which are better than TM hits to post-editors. In this framework, post-editors still work with the TM while benefiting from (better) SMT outputs; the assets in TMs are not wasted and TM fuzzy match scores can still be used to estimate (the upper bound of) post-editing labor.

There are three specific goals we need to achieve within this framework. Firstly, the recommendation should have high precision, otherwise it would be confusing for post-editors and may negatively affect the lower bound of the post-editing effort. Secondly, although we have full access to the SMT system used in this paper, our method should be able to generalize to cases where SMT is treated as a black-box, which is of-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 622–630,
Uppsala, Sweden, 11-16 July 2010.

ten the case in the translation industry. Finally, post-editors should be able to easily adjust the recommendation threshold to particular requirements without having to retrain the model.

In our framework, we recast translation recommendation as a binary classification (rather than regression) problem using SVMs, perform RBF kernel parameter optimization, employ posterior probability-based confidence estimation to support user-based tuning for precision and recall, experiment with feature sets involving MT-, TM- and system-independent features, and use automatic MT evaluation metrics to simulate post-editing effort.

The rest of the paper is organized as follows: we first briefly introduce related research in Section 2, and review the classification SVMs in Section 3. We formulate the classification model in Section 4 and present experiments in Section 5. In Section 6, we analyze the post-editing effort approximated by the TER metric (Snover et al., 2006). Section 7 concludes the paper and points out avenues for future research.

2 Related Work

Previous research relating to this work mainly focuses on predicting the MT quality.

The first strand is confidence estimation for MT, initiated by (Ueffing et al., 2003), in which posterior probabilities on the word graph or N-best list are used to estimate the quality of MT outputs. The idea is explored more comprehensively in (Blatz et al., 2004). These estimations are often used to rerank the MT output and to optimize it directly. Extensions of this strand are presented in (Quirk, 2004) and (Ueffing and Ney, 2005). The former experimented with confidence estimation with several different learning algorithms; the latter uses word-level confidence measures to determine whether a particular translation choice should be accepted or rejected in an interactive translation system.

The second strand of research focuses on combining TM information with an SMT system, so that the SMT system can produce better target language output when there is an exact or close match in the TM (Simard and Isabelle, 2009). This line of research is shown to help the performance of MT, but is less relevant to our task in this paper.

A third strand of research tries to incorporate confidence measures into a post-editing environment. To the best of our knowledge, the first paper in this area is (Specia et al., 2009a). Instead of modeling on translation quality (often measured by automatic evaluation scores), this research uses regression on both the automatic scores and scores assigned by post-editors. The method is improved in (Specia et al., 2009b), which applies Inductive Confidence Machines and a larger set of features to model post-editors' judgement of the translation quality between 'good' and 'bad', or among three levels of post-editing effort.

Our research is more similar in spirit to the third strand. However, we use outputs and features from the TM explicitly; therefore instead of having to solve a regression problem, we only have to solve a much easier binary prediction problem which can be integrated into TMs in a straightforward manner. Because of this, the precision and recall scores reported in this paper are not directly comparable to those in (Specia et al., 2009b) as the latter are computed on a pure SMT system without a TM in the background.

3 Support Vector Machines for Translation Quality Estimation

SVMs (Cortes and Vapnik, 1995) are binary classifiers that classify an input instance based on decision rules which minimize the regularized error function in (1):

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\limits_{i=1}^{l}\xi_i \\ \text{s. t.} & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{array} \quad (1)$$

where $(\mathbf{x}_i, y_i) \in R^n \times \{+1, -1\}$ are l training instances that are mapped by the function ϕ to a higher dimensional space. $\mathbf{w}$ is the weight vector, ξ is the relaxation variable and $C > 0$ is the penalty parameter.

Solving SVMs is viable using the 'kernel trick': finding a kernel function K in (1) with $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. We perform our experiments with the Radial Basis Function (RBF) kernel, as in (2):

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2), \gamma > 0 \quad (2)$$

When using SVMs with the RBF kernel, we have two free parameters to tune on: the cost parameter C in (1) and the radius parameter γ in (2).

In each of our experimental settings, the parameters C and γ are optimized by a brute-force grid

search. The classification result of each set of parameters is evaluated by cross validation on the training set.

4 Translation Recommendation as Binary Classification

We use an SVM binary classifier to predict the relative quality of the SMT output to make a recommendation. The SVM classifier uses features from the SMT system, the TM and additional linguistic features to estimate whether the SMT output is better than the hit from the TM.

4.1 Problem Formulation

As we treat translation recommendation as a binary classification problem, we have a pair of outputs from TM and MT for each sentence. Ideally the classifier will recommend the output that needs less post-editing effort. As large-scale annotated data is not yet available for this task, we use automatic TER scores (Snover et al., 2006) as the measure for the required post-editing effort. In the future, we hope to train our system on HTER (TER with human targeted references) scores (Snover et al., 2006) once the necessary human annotations are in place. In the meantime we use TER, as TER is shown to have high correlation with HTER.

We label the training examples as in (3):

$$y = \begin{cases} +1 & \text{if } TER(\text{MT}) < TER(\text{TM}) \\ -1 & \text{if } TER(\text{MT}) \geq TER(\text{TM}) \end{cases} \quad (3)$$

Each instance is associated with a set of features from both the MT and TM outputs, which are discussed in more detail in Section 4.3.

4.2 Recommendation Confidence Estimation

In classical settings involving SVMs, confidence levels are represented as margins of binary predictions. However, these margins provide little insight for our application because the numbers are only meaningful when compared to each other. What is more preferable is a probabilistic confidence score (e.g. 90% confidence) which is better understood by post-editors and translators.

We use the techniques proposed by (Platt, 1999) and improved by (Lin et al., 2007) to obtain the posterior probability of a classification, which is used as the confidence score in our system.

Platt's method estimates the posterior probability with a sigmod function, as in (4):

$$Pr(y = 1|\mathbf{x}) \approx P_{A,B}(f) \equiv \frac{1}{1 + exp(Af + B)} \quad (4)$$

where $f = f(\mathbf{x})$ is the decision function of the estimated SVM. A and B are parameters that minimize the cross-entropy error function F on the training data, as in Eq. (5):

$$\min_{z=(A,B)} F(z) = -\sum_{i=1}^{l}(t_i log(p_i) + (1 - t_i) log(1 - p_i)),$$
$$\text{where } p_i = P_{A,B}(f_i), \text{and } t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} \text{ if } y_i = +1 \\ \frac{1}{N_- + 2} \text{ if } y_i = -1 \end{cases} \quad (5)$$

where $z = (A, B)$ is a parameter setting, and N_+ and N_- are the numbers of observed positive and negative examples, respectively, for the label y_i. These numbers are obtained using an internal cross-validation on the training set.

4.3 The Feature Set

We use three types of features in classification: the MT system features, the TM feature and system-independent features.

4.3.1 The MT System Features

These features include those typically used in SMT, namely the phrase-translation model scores, the language model probability, the distance-based reordering score, the lexicalized reordering model scores, and the word penalty.

4.3.2 The TM Feature

The TM feature is the fuzzy match (Sikes, 2007) cost of the TM hit. The calculation of fuzzy match score itself is one of the core technologies in TM systems and varies among different vendors. We compute fuzzy match cost as the minimum Edit Distance (Levenshtein, 1966) between the source and TM entry, normalized by the length of the source as in (6), as most of the current implementations are based on edit distance while allowing some additional flexible matching.

$$h_{fm}(t) = \min_{e} \frac{EditDistance(s, e)}{Len(s)} \quad (6)$$

where s is the source side of t, the sentence to translate, and e is the source side of an entry in the TM. For fuzzy match scores F, this fuzzy match cost h_{fm} roughly corresponds to $1-F$. The difference in calculation does not influence classification, and allows direct comparison between a pure TM system and a translation recommendation system in Section 5.4.2.

4.3.3 System-Independent Features

We use several features that are independent of the translation system, which are useful when a third-party translation service is used or the MT system is simply treated as a black-box. These features are source and target side LM scores, pseudo source fuzzy match scores and IBM model 1 scores.

Source-Side Language Model Score and Perplexity. We compute the language model (LM) score and perplexity of the input source sentence on a LM trained on the source-side training data of the SMT system. The inputs that have lower perplexity or higher LM score are more similar to the dataset on which the SMT system is built.

Target-Side Language Model Perplexity. We compute the LM probability and perplexity of the target side as a measure of fluency. Language model perplexity of the MT outputs are calculated, and LM probability is already part of the MT systems scores. LM scores on TM outputs are also computed, though they are not as informative as scores on the MT side, since TM outputs should be grammatically perfect.

The Pseudo-Source Fuzzy Match Score. We translate the output back to obtain a pseudo source sentence. We compute the fuzzy match score between the original source sentence and this pseudo-source. If the MT/TM system performs well enough, these two sentences should be the same or very similar. Therefore, the fuzzy match score here gives an estimation of the confidence level of the output. We compute this score for both the MT output and the TM hit.

The IBM Model 1 Score. The fuzzy match score does not measure whether the hit could be a correct translation, i.e. it does not take into account the correspondence between the source and target, but rather only the source-side information. For the TM hit, the IBM Model 1 score (Brown et al., 1993) serves as a rough estimation of how good a translation it is on the word level; for the MT output, on the other hand, it is a black-box feature to estimate translation quality when the information from the translation model is not available. We compute bidirectional (source-to-target and target-to-source) model 1 scores on both TM and MT outputs.

5 Experiments

5.1 Experimental Settings

Our raw data set is an English–French translation memory with technical translation from Symantec, consisting of 51K sentence pairs. We randomly selected 43K to train an SMT system and translated the English side of the remaining 8K sentence pairs. The average sentence length of the training set is 13.5 words and the size of the training set is comparable to the (larger) TMs used in the industry. Note that we remove the exact matches in the TM from our dataset, because exact matches will be reused and not presented to the post-editor in a typical TM setting.

As for the SMT system, we use a standard log-linear PB-SMT model (Och and Ney, 2002): GIZA++ implementation of IBM word alignment model 4,[1] the refinement and phrase-extraction heuristics described in (Koehn et al., 2003), minimum-error-rate training (Och, 2003), a 5-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995) trained with SRILM (Stolcke, 2002) on the English side of the training data, and Moses (Koehn et al., 2007) to decode. We train a system in the opposite direction using the same data to produce the pseudo-source sentences.

We train the SVM classifier using the libSVM (Chang and Lin, 2001) toolkit. The SVM-training and testing is performed on the remaining 8K sentences with 4-fold cross validation. We also report 95% confidence intervals.

The SVM hyper-parameters are tuned using the training data of the first fold in the 4-fold cross validation via a brute force grid search. More specifically, for parameter C in (1) we search in the range $[2^{-5}, 2^{15}]$, and for parameter γ (2) we search in the range $[2^{-15}, 2^{3}]$. The step size is 2 on the exponent.

5.2 The Evaluation Metrics

We measure the quality of the classification by precision and recall. Let A be the set of recommended MT outputs, and B be the set of MT outputs that have lower TER than TM hits. We standardly define precision P, recall R and F-value as in (7):

[1] More specifically, we performed 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

$$P = \frac{|A \bigcap B|}{|A|}, R = \frac{|A \bigcap B|}{|B|} \text{ and } F = \frac{2PR}{P+R} \quad (7)$$

5.3 Recommendation Results

In Table 1, we report recommendation performance using MT and TM system features (SYS), system features plus system-independent features (ALL:SYS+SI), and system-independent features only (SI).

Table 1: Recommendation Results

	Precision	Recall	F-Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
SI	82.56±1.46	95.83±0.52	88.70±.65
ALL	83.45±1.33	95.56±1.33	89.09±.24

From Table 1, we observe that MT and TM system-internal features are very useful for producing a stable (as indicated by the smaller confidence interval) recommendation system (SYS). Interestingly, only using some simple system-external features as described in Section 4.3.3 can also yield a system with reasonably good performance (SI). We expect that the performance can be further boosted by adding more syntactic and semantic features. Combining all the system-internal and -external features leads to limited gains in Precision and F-score compared to using only system-internal features (SYS) only. This indicates that at the default confidence level, current system-external (resp. system-internal) features can only play a limited role in informing the system when current system-internal (resp. system-external) features are available. We show in Section 5.4.2 that combing both system-internal and -external features can yield higher, more stable precision when adjusting the confidence levels of the classifier. Additionally, the performance of system SI is promising given the fact that we are using only a limited number of simple features, which demonstrates a good prospect of applying our recommendation system to MT systems where we do not have access to their internal features.

5.4 Further Improving Recommendation Precision

Table 1 shows that classification recall is very high, which suggests that precision can still be improved, even though the F-score is not low. Considering that TM is the dominant technology used by post-editors, a recommendation to replace the hit from the TM would require more confidence, i.e. higher precision. Ideally our aim is to obtain a level of 0.9 precision at the cost of some recall, if necessary. We propose two methods to achieve this goal.

5.4.1 Classifier Margins

We experiment with different margins on the training data to tune precision and recall in order to obtain a desired balance. In the basic case, the training example would be marked as in (3). If we label both the training and test sets with this rule, the accuracy of the prediction will be maximized.

We try to achieve higher precision by enforcing a larger bias towards negative examples in the training set so that some borderline positive instances would actually be labeled as negative, and the classifier would have higher precision in the prediction stage as in (8).

$$y = \begin{cases} +1 & \text{if } TER(\text{SMT}) + b < TER(\text{TM}) \\ -1 & \text{if } TER(\text{SMT}) + b \geqslant TER(\text{TM}) \end{cases} \quad (8)$$

We experiment with b in [0, 0.25] using MT system features and TM features. Results are reported in Table 2.

Table 2: Classifier margins

	Precision	Recall
TER+0	83.45±1.33	95.56±1.33
TER+0.05	82.41±1.23	94.41±1.01
TER+0.10	84.53±0.98	88.81±0.89
TER+0.15	85.24±0.91	87.08±2.38
TER+0.20	87.59±0.57	75.86±2.70
TER+0.25	89.29±0.93	66.67±2.53

The highest accuracy and F-value is achieved by $TER + 0$, as all other settings are trained on biased margins. Except for a small drop in $TER{+}0.05$, other configurations all obtain higher precision than $TER + 0$. We note that we can obtain 0.85 precision without a big sacrifice in recall with b=0.15, but for larger improvements on precision, recall will drop more rapidly.

When we use b beyond 0.25, the margin becomes less reliable, as the number of positive examples becomes too small. In particular, this causes the SVM parameters we tune on in the first fold to become less applicable to the other folds. This is one limitation of using biased margins to

obtain high precision. The method presented in Section 5.4.2 is less influenced by this limitation.

5.4.2 Adjusting Confidence Levels

An alternative to using a biased margin is to output a confidence score during prediction and to threshold on the confidence score. It is also possible to add this method to the SVM model trained with a biased margin.

We use the SVM confidence estimation techniques in Section 4.2 to obtain the confidence level of the recommendation, and change the confidence threshold for recommendation when necessary. This also allows us to compare directly against a simple baseline inspired by TM users. In a TM environment, some users simply ignore TM hits below a certain fuzzy match score F (usually from 0.7 to 0.8). This fuzzy match score reflects the confidence of recommending the TM hits. To obtain the confidence of recommending an SMT output, our baseline (FM) uses fuzzy match costs $h_{FM} \approx 1-F$ (cf. Section 4.3.2) for the TM hits as the level of confidence. In other words, the higher the fuzzy match cost of the TM hit is (lower fuzzy match score), the higher the confidence of recommending the SMT output. We compare this baseline with the three settings in Section 5.

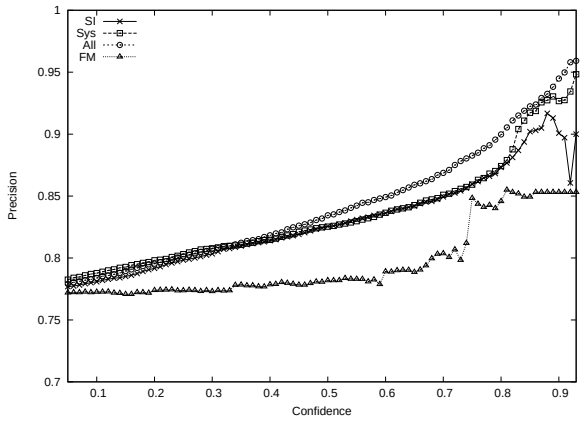


Figure 1: Precision Changes with Confidence Level

Figure 1 shows that the precision curve of FM is low and flat when the fuzzy match costs are low (from 0 to 0.6), indicating that it is unwise to recommend an SMT output when the TM hit has a low fuzzy match cost (corresponding to higher fuzzy match score, from 0.4 to 1). We also observe that the precision of the recommendation receives a boost when the fuzzy match costs for the TM hits are above 0.7 (fuzzy match score lower than 0.3), indicating that SMT output should be recommended when the TM hit has a high fuzzy match cost (low fuzzy match score). With this boost, the precision of the baseline system can reach 0.85, demonstrating that a proper thresholding of fuzzy match scores can be used effectively to discriminate the recommendation of the TM hit from the recommendation of the SMT output.

However, using the TM information only does not always find the easiest-to-edit translation. For example, an excellent SMT output should be recommended even if there exists a good TM hit (e.g. fuzzy match score is 0.7 or more). On the other hand, a misleading SMT output should not be recommended if there exists a poor but useful TM match (e.g. fuzzy match score is 0.2).

Our system is able to tackle these complications as it incorporates features from the MT and the TM systems simultaneously. Figure 1 shows that both the SYS and the ALL setting consistently outperform FM, indicating that our classification scheme can better integrate the MT output into the TM system than this naive baseline.

The SI feature set does not perform well when the confidence level is set above 0.85 (cf. the descending tail of the SI curve in Figure 1). This might indicate that this feature set is not reliable enough to extract the best translations. However, when the requirement on precision is not that high, and the MT-internal features are not available, it would still be desirable to obtain translation recommendations with these black-box features. The difference between SYS and ALL is generally small, but ALL performs steadily better in [0.5, 0,8].

Table 3: Recall at Fixed Precision

	Recall
SYS @85PREC	88.12±1.32
SYS @90PREC	52.73±2.31
SI @85PREC	87.33±1.53
ALL @85PREC	88.57±1.95
ALL @90PREC	51.92±4.28

5.5 Precision Constraints

In Table 3 we also present the recall scores at 0.85 and 0.9 precision for SYS, SI and ALL models to demonstrate our system's performance when there is a hard constraint on precision. Note that our system will return the TM entry when there is an exact match, so the overall precision of the system

is above the precision score we set here in a mature TM environment, as a significant portion of the material to be translated will have a complete match in the TM system.

In Table 3 for MODEL@K, the recall scores are achieved when the prediction precision is better than K with 0.95 confidence. For each model, precision at 0.85 can be obtained without a very big loss on recall. However, if we want to demand further recommendation precision (more conservative in recommending SMT output), the recall level will begin to drop more quickly. If we use only system-independent features (SI), we cannot achieve as high precision as with other models even if we sacrifice more recall.

Based on these results, the users of the TM system can choose between precision and recall according to their own needs. As the threshold does not involve training of the SMT system or the SVM classifier, the user is able to determine this trade-off at runtime.

Table 4: Contribution of Features

	Precision	Recall	F Score
SYS	82.53±1.17	96.44±0.68	88.95±.56
+M1	82.87±1.26	96.23±0.53	89.05±.52
+LM	82.82±1.16	96.20±1.14	89.01±.23
+PS	83.21±1.33	96.61±0.44	89.41±.84

5.6 Contribution of Features

In Section 4.3.3 we suggested three sets of system-independent features: features based on the source- and target-side language model (LM), the IBM Model 1 (M1) and the fuzzy match scores on pseudo-source (PS). We compare the contribution of these features in Table 4.

In sum, all the three sets of system-independent features improve the precision and F-scores of the MT and TM system features. The improvement is not significant, but improvement on every set of system-independent features gives some credit to the capability of SI features, as does the fact that SI features perform close to SYS features in Table 1.

6 Analysis of Post-Editing Effort

A natural question on the integration models is whether the classification reduces the effort of the translators and post-editors: after reading these recommendations, will they translate/edit less than they would otherwise have to? Ideally this question would be answered by human post-editors in a large-scale experimental setting. As we have not yet conducted a manual post-editing experiment, we conduct two sets of analyses, trying to show which type of edits will be required for different recommendation confidence levels. We also present possible methods for human evaluation at the end of this section.

6.1 Edit Statistics

We provide the statistics of the number of edits for each sentence with 0.95 confidence intervals, sorted by TER edit types. Statistics of positive instances in classification (i.e. the instances in which MT output is recommended over the TM hit) are given in Table 5.

When an MT output is recommended, its TM counterpart will require a larger average number of total edits than the MT output, as we expect. If we drill down, however, we also observe that many of the saved edits come from the *Substitution* category, which is the most costly operation from the post-editing perspective. In this case, the recommended MT output actually saves more effort for the editors than what is shown by the TER score. It reflects the fact that TM outputs are not actual translations, and might need heavier editing.

Table 6 shows the statistics of negative instances in classification (i.e. the instances in which MT output is not recommended over the TM hit). In this case, the MT output requires considerably more edits than the TM hits in terms of all four TER edit types, i.e. insertion, substitution, deletion and shift. This reflects the fact that some high quality TM matches can be very useful as a translation.

6.2 Edit Statistics on Recommendations of Higher Confidence

We present the edit statistics of recommendations with higher confidence in Table 7. Comparing Tables 5 and 7, we see that if recommended with higher confidence, the MT output will need substantially less edits than the TM output: e.g. 3.28 fewer substitutions on average.

From the characteristics of the high confidence recommendations, we suspect that these mainly comprise harder to translate (i.e. different from the SMT training set/TM database) sentences, as indicated by the slightly increased edit operations

Table 5: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	0.9849 ± 0.0408	2.2881 ± 0.0672	0.8686 ± 0.0370	1.2500 ± 0.0598
TM	0.7762 ± 0.0408	4.5841 ± 0.1036	3.1567 ± 0.1120	1.2096 ± 0.0554

Table 6: Edit Statistics when NOT Recommending MT Outputs in Classification, confidence=0.5

	Insertion	Substitution	Deletion	Shift
MT	1.0830 ± 0.1167	2.2885 ± 0.1376	1.0964 ± 0.1137	1.5381 ± 0.1962
TM	0.7554 ± 0.0376	1.5527 ± 0.1584	1.0090 ± 0.1850	0.4731 ± 0.1083

Table 7: Edit Statistics when Recommending MT Outputs in Classification, confidence=0.85

	Insertion	Substitution	Deletion	Shift
MT	1.1665 ± 0.0615	2.7334 ± 0.0969	1.0277 ± 0.0544	1.5549 ± 0.0899
TM	0.8894 ± 0.0594	6.0085 ± 0.1501	4.1770 ± 0.1719	1.6727 ± 0.0846

on the MT side. TM produces much worse edit-candidates for such sentences, as indicated by the numbers in Table 7, since TM does not have the ability to automatically reconstruct an output through the combination of several segments.

6.3 Plan for Human Evaluation

Evaluation with human post-editors is crucial to validate and improve translation recommendation. There are two possible avenues to pursue:

- Test our system on professional post-editors. By providing them with the TM output, the MT output and the one recommended to edit, we can measure the true accuracy of our recommendation, as well as the post-editing time we save for the post-editors;
- Apply the presented method on open domain data and evaluate it using crowdsourcing. It has been shown that crowdsourcing tools, such as the Amazon Mechanical Turk (Callison-Burch, 2009), can help developers to obtain good human judgements on MT output quality both cheaply and quickly. Given that our problem is related to MT quality estimation in nature, it can potentially benefit from such tools as well.

7 Conclusions and Future Work

In this paper we present a classification model to integrate SMT into a TM system, in order to facilitate the work of post-editors. Insodoing we handle the problem of MT quality estimation as binary prediction instead of regression. From the post-editors' perspective, they can continue to work in their familiar TM environment, use the same cost-estimation methods, and at the same time benefit from the power of state-of-the-art MT. We use SVMs to make these predictions, and use grid search to find better RBF kernel parameters.

We explore features from inside the MT system, from the TM, as well as features that make no assumption on the translation model for the binary classification. With these features we make glass-box and black-box predictions. Experiments show that the models can achieve 0.85 precision at a level of 0.89 recall, and even higher precision if we sacrifice more recall. With this guarantee on precision, our method can be used in a TM environment without changing the upper-bound of the related cost estimation.

Finally, we analyze the characteristics of the integrated outputs. We present results to show that, if measured by number, type and content of edits in TER, the recommended sentences produced by the classification model would bring about less post-editing effort than the TM outputs.

This work can be extended in the following ways. Most importantly, it is useful to test the model in user studies, as proposed in Section 6.3. A user study can serve two purposes: 1) it can validate the effectiveness of the method by measuring the amount of edit effort it saves; and 2) the byproduct of the user study – post-edited sentences – can be used to generate HTER scores to train a better recommendation model. Furthermore, we want to experiment and improve on the adaptability of this method, as the current experiment is on a specific domain and language pair.

Acknowledgements

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University. We thank Symantec for providing the TM database and the anonymous reviewers for their insightful comments.

References

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *The 20th International Conference on Computational Linguistics (Coling-2004)*, pages 315 – 321, Geneva, Switzerland.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263 – 311.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *The 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*, pages 286 – 295, Singapore.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273 – 297.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *The 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181 – 184, Detroit, MI.

Philipp. Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *The 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL/HLT-2003)*, pages 48 – 54, Edmonton, Alberta, Canada.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *The 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions (ACL-2007)*, pages 177 – 180, Prague, Czech Republic.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707 – 710.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. 2007. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267 – 276.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *The 41st Annual Meeting on Association for Computational Linguistics (ACL-2003)*, pages 160 – 167.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61 – 74.

Christopher B. Quirk. 2004. Training a sentence-level machine translation confidence measure. In *The Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, pages 825 – 828, Lisbon, Portugal.

Richard Sikes. 2007. Fuzzy matching in theory and practice. *Multilingual*, 18(6):39 – 43.

Michel Simard and Pierre Isabelle. 2009. Phrase-based machine translation in a computer-assisted translation environment. In *The Twelfth Machine Translation Summit (MT Summit XII)*, pages 120 – 127, Ottawa, Ontario, Canada.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *The 2006 conference of the Association for Machine Translation in the Americas (AMTA-2006)*, pages 223 – 231, Cambridge, MA.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009a. Estimating the sentence-level quality of machine translation systems. In *The 13th Annual Conference of the European Association for Machine Translation (EAMT-2009)*, pages 28 – 35, Barcelona, Spain.

Lucia Specia, Craig Saunders, Marco Turchi, Zhuoran Wang, and John Shawe-Taylor. 2009b. Improving the confidence of machine translation quality estimates. In *The Twelfth Machine Translation Summit (MT Summit XII)*, pages 136 – 143, Ottawa, Ontario, Canada.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *The Seventh International Conference on Spoken Language Processing*, volume 2, pages 901 – 904, Denver, CO.

Nicola Ueffing and Hermann Ney. 2005. Application of word-level confidence measures in interactive statistical machine translation. In *The Ninth Annual Conference of the European Association for Machine Translation (EAMT-2005)*, pages 262 – 270, Budapest, Hungary.

Nicola Ueffing, Klaus Macherey, and Hermann Ney. 2003. Confidence measures for statistical machine translation. In *The Ninth Machine Translation Summit (MT Summit IX)*, pages 394 – 401, New Orleans, LA.

On Jointly Recognizing and Aligning Bilingual Named Entities

Yufeng Chen, Chengqing Zong
Institute of Automation, Chinese Academy of Sciences
Beijing, China
{chenyf,cqzong}@nlpr.ia.ac.cn

Keh-Yih Su
Behavior Design Corporation
Hsinchu, Taiwan, R.O.C.
bdc.kysu@gmail.com

Abstract

We observe that (1) how a given named entity (NE) is translated (i.e., either semantically or phonetically) depends greatly on its associated entity type, and (2) entities within an aligned pair should share the same type. Also, (3) those initially detected NEs are anchors, whose information should be used to give certainty scores when selecting candidates. From this basis, an integrated model is thus proposed in this paper to jointly identify and align bilingual named entities between Chinese and English. It adopts a new mapping type ratio feature (which is the proportion of NE internal tokens that are semantically translated), enforces an entity type consistency constraint, and utilizes additional monolingual candidate certainty factors (based on those NE anchors). The experiments show that this novel approach has substantially raised the type-sensitive F-score of identified NE-pairs from 68.4% to 81.7% (42.1% F-score imperfection reduction) in our Chinese-English NE alignment task.

1 Introduction

In trans-lingual language processing tasks, such as machine translation and cross-lingual information retrieval, *named entity* (NE) translation is essential. Bilingual NE alignment, which links source NEs and target NEs, is the first step to train the NE translation model.

Since NE alignment can only be conducted after its associated NEs have first been identified, the including-rate of the first recognition stage significantly limits the final alignment performance. To alleviate the above error accumulation problem, two strategies have been proposed in the literature. The first strategy (Al-Onaizan and Knight, 2002; Moore, 2003; Feng et al., 2004; Lee et al., 2006) identifies NEs only on the source side and then finds their corresponding NEs on the target side. In this way, it avoids the NE recognition errors which would otherwise be brought into the alignment stage from the target side; however, the NE errors from the source side still remain.

To further reduce the errors from the source side, the second strategy (Huang et al., 2003) expands the NE candidate-sets in both languages before conducting the alignment, which is done by treating the original results as anchors, and then re-generating further candidates by enlarging or shrinking those anchors' boundaries. Of course, this strategy will be in vain if the NE anchor is missed in the initial detection stage. In our data-set, this strategy significantly raises the NE-pair type-insensitive including-rate [1] from 83.9% to 96.1%, and is thus adopted in this paper.

Although the above expansion strategy has substantially alleviated the error accumulation problem, the final alignment accuracy is still not good (type-sensitive F-score only 68.4%, as indicated in Table 2 in Section 4.2). After having examined the data, we found that: (1) How a given NE is translated, either semantically (called *translation*) or phonetically (called *transliteration*), depends greatly on its associated entity type[2]. The *mapping type ratio*, which is the percentage of NE internal tokens which are translated semantically, can help with the recognition of the associated NE type; (2) Entities within an aligned pair should share the same type, and this restriction should be integrated into NE alignment as a constraint; (3) Those initially identified monolingual NEs can act as anchors to give *monolingual candidate certainty scores*

[1] Which is the percentage of desired NE-pairs that are included in the expanded set, and is the upper bound on NE alignment performance (regardless of NE types).

[2] The proportions of semantic translation, which denote the ratios of semantically translated words among all the associated NE words, for person names (PER), location names (LOC), and organization names (ORG) approximates 0%, 28.6%, and 74.8% respectively in Chinese-English name entity list (2005T34) released by the Linguistic Data Consortium (LDC). Since the title, such as "sir" and "chairman", is not considered as a part of person names in this corpus, PERs are all transliterated there.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 631–639,
Uppsala, Sweden, 11-16 July 2010.

(preference weightings) for the re-generated candidates.

Based on the above observation, a new joint model which adopts the *mapping type ratio*, enforces the *entity type consistency* constraint, and also utilizes the *monolingual candidate certainty factors* is proposed in this paper to jointly identify and align bilingual NEs under an integrated framework. This framework is decomposed into three subtasks: *Initial Detection*, *Expansion*, and *Alignment&Re-identification*. The *Initial Detection* subtask first locates the initial NEs and their associated NE types inside both the Chinese and English sides. Afterwards, the *Expansion* subtask re-generates the candidate-sets in both languages to recover those initial NE recognition errors. Finally, the *Alignment&Re-identification* subtask *jointly* recognizes and aligns bilingual NEs via the proposed joint model presented in Section 3. With this new approach, 41.8% imperfection reduction in type-sensitive F-score, from 68.4% to 81.6%, has been observed in our Chinese-English NE alignment task.

2 Motivation

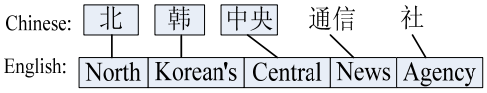
The problem of NE recognition requires both boundary identification and type classification. However, the complexity of these tasks varies with different languages. For example, Chinese NE boundaries are especially difficult to identify because Chinese is not a tokenized language. In contrast, English NE boundaries are easier to identify due to capitalization clues. On the other hand, classification of English NE types can be more challenging (Ji et al., 2006). Since alignment would force the linked NE pair to share the same semantic meaning, the NE that is more reliably identified in one language can be used to ensure its counterpart in another language. This benefits both the NE boundary identification and type classification processes, and it hints that alignment can help to re-identify those initially recognized NEs which had been less reliable.

As shown in the following example, although the desired NE "北韩中央通信社" is recognized partially as "北韩中央" in the initial recognition stage, it would be more preferred if its English counterpart "North Korean's Central News Agency" is given. The reason for this is that "News Agency" would prefer to be linked to "通信社", rather than to be deleted (which would happen if "北韩中央" is chosen as the corresponding Chinese NE).

(I) The initial NE detection in a Chinese sentence:
官方的 <ORG>北韩中央</ORG> 通信社引述海军...

(II) The initial NE detection of its English counterpart:
Official <ORG>North Korean's Central News Agency </ORG> quoted the navy's statement…

(III) The word alignment between two NEs:



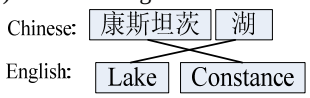
(VI) The re-identified Chinese NE boundary after alignment:
官方的 <ORG>北韩中央通信社</ORG> 引述海军声明...

As another example, the word "lake" in the English NE is linked to the Chinese character "湖" as illustrated below, and this mapping is found to be a translation and not a transliteration. Since translation rarely occurs for personal names (Chen et al., 2003), the desired NE type "LOC" would be preferred to be shared between the English NE "Lake Constance" and its corresponding Chinese NE "康斯坦茨湖". As a result, the original incorrect type "PER" of the given English NE is fixed, and the necessity of using mapping type ratio and NE type consistency constraint becomes evident.

(I) The initial NE detection result in a Chinese sentence:
在 <LOC>康斯坦茨湖</LOC> 工作的一艘渡船船长…

(II) The initial NE detection of its English counterpart:
The captain of a ferry boat who works on <PER>Lake Constance </PER>…

(III) The word alignment between two NEs:



(VI) The re-identified English NE type after alignment:
The captain of a ferry boat who works on <LOC>Lake Constance</LOC>…

3 The Proposed Model

As mentioned in the introduction section, given a Chinese-English sentence-pair (CS, ES), with its initially recognized Chinese NEs $< CNE_i, CType_i >_{i=1}^{S}, S \geq 1$ and English NEs $[ENE_j, EType_j]_{j=1}^{T}, T \geq 1$ ($CType_i$ and $Etype_j$ are original NE types assigned to CNE_i and ENE_j, respectively), we will first re-generate two NE candidate-sets from them by enlarging and shrinking the boundaries of those initially recognized NEs. Let $RCNE_1^{K_C}$ and $RENE_1^{K_E}$ denote these two *re-generated* candidate sets for Chinese and English NEs respectively (K_C and K_E are their set-sizes), and $K = \min(S, T)$, then a total K pairs of final Chinese and English NEs will be picked up from the Cartesian product of

$RCNE_1^{K_C}$ and $RENE_1^{K_E}$, according to their associated *linking score*, which is defined as follows.

Let $Score(RCNE_{<k>}, RENE_{[k]})$ denote the associated *linking score* for a given candidate-pair $RCNE_{<k>}$ and $RENE_{[k]}$, where $<k>$ and $[k]$ are the associated indexes of the re-generated Chinese and English NE candidates, respectively. Furthermore, let $RType_k$ be the NE type to be *re-assigned* and shared by $RCNE_{<k>}$ and $RENE_{[k]}$ (as they possess the same meaning). Assume that $RCNE_{<k>}$ and $RENE_{[k]}$ are derived from initially recognized CNE_i and ENE_j, respectively, and M_{IC} denotes their *internal component mapping*, to be defined in Section 3.1, then $Score(RCNE_{<k>}, RENE_{[k]})$ is defined as follows:

$$Score(RCNE_{<k>}, RENE_{[k]}) = \max_{M_{IC}, RType_k} P\begin{pmatrix} M_{IC}, RType_k, RCNE_{<k>}, RENE_{[k]} \mid \\ <CNE_i, CType_i>, CS, [ENE_j, EType_j], ES \end{pmatrix} \quad (1)$$

Here, the "max" operator varies over each possible internal component mapping M_{IC} and re-assigned type (PER, LOC, and ORG). For brevity, we will drop those associated subscripts from now on, if there is no confusion.

The associated probability factors in the above linking score can be further derived as follows.

$$\begin{aligned} & P\left(M_{IC}, RType, RCNE, RENE \,\middle|\, \begin{matrix} <CNE, CType>, CS, \\ [ENE, EType], ES \end{matrix} \right) \\ & \cong P(M_{IC} \mid RType, RCNE, RENE) \\ & \quad \times P(RCNE \mid CNE, CType, CS, RType) \\ & \quad \times P(RENE \mid ENE, EType, ES, RType) \\ & \quad \times P(RType \mid CNE, ENE, CType, EType) \end{aligned} \quad (2)$$

In the above equation, $P(M_{IC} \mid RType, RCNE, RENE)$ and $P(RType \mid CNE, ENE, CType, EType)$ are the *Bilingual Alignment Factor* and the *Bilingual Type Re-assignment Factor* respectively, to represent the bilingual related scores (Section 3.1). Also, $P(RCNE \mid CNE, CType, CS, RType)$ and $P(RENE \mid ENE, EType, ES, RType)$ are *Monolingual Candidate Certainty Factors* (Section 3.2) used to assign preference to each selected *RCNE* and *RENE*, based on the initially recognized NEs (which act as anchors).

3.1 Bilingual Related Factors

The *bilingual alignment factor* mainly represents the likelihood value of a specific internal component mapping M_{IC}, given a pair of possible NE configurations *RCNE* and *RENE* and their associated *RType*. Since Chinese word segmentation is problematic, especially for transliterated words, the bilingual alignment factor $P(M_{IC} \mid RType, RCNE, RENE)$ in Eq (2) is derived to be conditioned on *RENE* (i.e., starting from the English part).

We define the internal component mapping M_{IC} to be $M_{IC} \equiv <[cpn_{<n>}, ew_{[n]}, Mtype_n]_{n=1}^{N}, \delta>$, where $[cpn_{<n>}, ew_{[n]}, Mtype_n]$ denotes a linked pair consisting of a *Chinese component* $cpn_{<n>}$ (which might contain several Chinese characters) and an *English word* $ew_{[n]}$ within *RCNE* and *RENE* respectively, with their *internal mapping type* $Mtype_n$ to be either *translation* (abbreviated as *TS*) or *transliteration* (abbreviated as *TL*). In total, there are N component mappings, with N_{TS} translation mappings $[cpn_{<n_1>}, ew_{[n_1]}, TS]_{n_1=1}^{N_{TS}}$ and N_{TL} transliteration mappings $[cpn_{<n_2>}, ew_{[n_2]}, TL]_{n_2=1}^{N_{TL}}$, so that $N = N_{TS} + N_{TL}$.

Moreover, since the mapping type distributions of various NE types deviate greatly from one another, as illustrated in the second footnote, the associated *mapping type ratio* $\delta = (N_{TS} / N)$ is thus an important feature, and is included in the internal component mapping configuration specified above. For example, the M_{IC} between "康斯坦茨湖" and "Constance Lake" is [康斯坦茨, *Constance*, *TL*] and [湖, *Lake*, *TS*], so its associated mapping type ratio will be "0.5" (i.e., 1/2). Therefore, the internal mapping $P(M_{IC} \mid RType, RENE)$ is further deduced by introducing the internal mapping type $Mtype_n$ and the mapping type ratio δ as follows:

$$\begin{aligned} & P(M_{IC} \mid RType, RENE) \\ & \equiv P([cpn_{<n>}, ew_{[n]}, Mtype_n]_{n=1}^{N}, \delta \mid RType, RENE) \\ & \approx \prod_{n=1}^{N} \begin{bmatrix} P(cpn_{<n>} \mid Mtype_n, ew_{[n]}, RType) \\ \times P(Mtype_n \mid ew_{[n]}, RType) \end{bmatrix} \\ & \quad \times P(\delta \mid RType) \end{aligned} \quad (3)$$

In the above equation, the mappings between internal components are trained from the syllable/word alignment of NE pairs of different NE types. In more detail, for transliteration, the model adopted in (Huang et al., 2003), which first Romanizes Chinese characters and then transliterates them into English characters, is

used for $P(cpn_{<n>} | TL_n, ew_{[n]}, RType)$. For translation, conditional probability is directly used for $P(cpn_{<n>} | TS_n, ew_{[n]}, RType)$.

Lastly, the *bilingual type re-assignment factor* $P(RType | CNE, ENE, CType, EType)$ proposed in Eq (2) is derived as follows:

$$P(RType | RCNE, RENE, CType, EType) \cong P(RType | CType, EType) \quad (4)$$

As Eq (4) shows, both the Chinese initial NE type and English initial NE type are adopted to jointly identify their shared NE type *RType*.

3.2 Monolingual Candidate Certainty Factors

On the other hand, the *monolingual candidate certainty factors* in Eq (2) indicate the likelihood that a re-generated NE candidate is the true NE given its originally detected NE. For Chinese, it is derived as follows:

$$\begin{aligned} & P(RCNE | CNE, CType, CS, RType) \\ & \cong P(LeftD, RightD, Str[RCNE] | Len_C, CType, RType) \\ & \approx P(LeftD | Len_C, CType, RType) \\ & \times P(RightD | Len_C, CType, RType) \\ & \times \prod_{m=1}^{M} P(cc_m | cc_{m-1}, RType) \end{aligned} \quad (5)$$

Where, the subscript *C* denotes Chinese, and Len_C is the *length* of the originally recognized Chinese NE *CNE*. *LeftD* and *RightD* denote the *left and right distance* (which are the numbers of Chinese characters) that *RCNE* shrinks/enlarges from the left and right boundary of its anchor *CNE*, respectively. As in the above example, assume that *CNE* and *RCNE* are "北韩中央" and "韩中央通信社" respectively, *LeftD* and *RightD* will be "-1" and "+3". Also, *Str*[*RCNE*] stands for the associated Chinese string of *RCNE*, cc_m denotes the *m-th Chinese character* within that string, and *M* denotes the total number of Chinese characters within *RCNE*.

On the English side, following Eq (5), $P(RENE | ENE, EType, ES, RType)$ can be derived similarly, except that *LeftD* and *RightD* will be measured in number of English words. For instance, with *ENE* and *RENE* as "*Lake Constance*" and "*on Lake Constance*" respectively, *LeftD* and *RightD* will be "+1" and "0". Also, the bigram unit cc_m of the Chinese NE string is replaced by the English word unit ew_n.

All the bilingual and monolingual factors mentioned above, which are derived from Eq (1), are weighted differently according to their contributions. The corresponding weighting coefficients are obtained using the well-known *Minimum Error Rate Training* (Och, 2003; commonly abbreviated as MERT) algorithm by minimizing the number of associated errors in the development set.

3.3 Framework for the Proposed Model

The above model is implemented with a three-stage framework: (A) Initial NE Recognition; (B) NE-Candidate-Set Expansion; and (C) NE Alignment&Re-identification. The Following Diagram gives the details of this framework:

For each given bilingual sentence-pair:

(A) Initial NE Recognition: generates the initial NE anchors with off-the-self packages.

(B) NE-Candidate-Set Expansion: For each initially detected NE, several NE candidates will be re-generated from the original NE by allowing its boundaries to be shrunk or enlarged within a pre-specified range.

- (B.1) Create both RCNE and RENE candidate-sets, which are expanded from those initial NEs identified in the previous stage.
- (B.2) Construct an NE-pair candidate-set (named *NE-Pair-Candidate-Set*), which is the Cartesian product of the RCNE and RENE candidate-sets created above.

(C) NE Alignment&Re-identification: Rank each candidate in the NE-Pair-Candidate-Set constructed above with the linking score specified in Eq (1). Afterwards, conduct a beam search process to select the top *K* non-overlapping NE-pairs from this set.

Diagram 1. Steps to Generate the Final NE-Pairs

It is our observation that, four Chinese characters for both shrinking and enlarging, two English words for shrinking and three for enlarging are enough in most cases. Under these conditions, the including-rates for NEs with correct boundaries are raised to 95.8% for Chinese and 97.4% for English; and even the NE-pair including rate is raised to 95.3%. Since the above range limitation setting has an including-rate only 0.8% lower than that can be obtained without any range limitation (which is 96.1%), it is adopted in this paper to greatly reduce the number of NE-pair-candidates.

4 Experiments

To evaluate the proposed joint approach, a prior work (Huang et al., 2003) is re-implemented in our environment as the baseline, in which the translation cost, transliteration cost and tagging cost are used. This model is selected for comparison because it not only adopts the same candidate-set expansion strategy as mentioned above, but also utilizes the monolingual information when selecting NE-pairs (however, only a simple bi-gram model is used as the tagging cost in their paper). Note that it enforces the same NE type only when the tagging cost is evaluated:

$$C_{tag} = \min_{RType}[-\log(\prod_{m=1}^{M} P(cc_m \mid cc_{m-1}, RType)) - \log(\prod_{n=1}^{N} P(ew_n \mid ew_{n-1}, RType))]$$

To give a fairer comparison, the same training-set and testing-set are adopted. The training-set includes two parts. The first part consists of 90,412 aligned sentence-pairs newswire data from the Foreign Broadcast Information Service (FBIS), which is denoted as Training-Set-I. The second Part of the training set is the LDC2005T34 bilingual NE dictionary[3], which is denoted as Training-Set-II. The required feature information is then manually labeled throughout the two training sets.

In our experiments, for the baseline system, the translation cost and the transliteration cost are trained on Training-Set-II, while the tagging cost is trained on Training-Set-I. For the proposed approach, the monolingual candidate certainty factors are trained on Training-Set-I, and Training-Set-II is used to train the parameters relating to bilingual alignment factors.

For the testing-set, 300 sentence pairs are randomly selected from the LDC Chinese-English News Text (LDC2005T06). The average length of the Chinese sentences is 59.4 characters, while the average length of the English sentences is 24.8 words. Afterwards, the answer keys for NE recognition and alignment were annotated manually, and used as the gold standard to calculate metrics of precision (P), recall (R), and F-score (F) for both *NE recognition* (NER) and *NE alignment* (NEA). In Total 765 Chinese NEs and 747 English NEs were manually labeled in the testing-set, within which there are only 718 NE pairs, including 214 PER, 371 LOC and 133 ORG NE-pairs. The number of NE pairs is less than that of NEs, because not all those recognized NEs can be aligned.

Besides, the development-set for MERT weight training is composed of 200 sentence pairs selected from the LDC2005T06 corpus, which includes 482 manually tagged NE pairs. There is no overlap between the training-sets, the development-set and the testing-set.

4.1 Baseline System

Both the baseline and the proposed models share the same initial detection subtask, which adopts the Chinese NE recognizer reported by Wu et al. (2005), which is a hybrid statistical model incorporating multi-knowledge sources, and the English NE recognizer included in the publicly available Mallet toolkit[4] to generate initial NEs. Initial Chinese NEs and English NEs are recognized by these two available packages respectively.

NE-type	P (%): C/E	R (%): C/E	F (%): C/E
PER	80.2 / 79.2	**87.7 / 85.3**	83.8 / 82.1
LOC	**89.8 / 85.9**	87.3 / 81.5	**88.5/ 83.6**
ORG	78.6 / 82.9	82.8 / 79.6	80.6 / 81.2
ALL	83.4 / 82.1	86.0 / 82.6	84.7 / 82.3

Table 1. Initial Chinese/English NER

Table 1 shows the initial NE recognition performances for both Chinese and English (the largest entry in each column is highlighted for visibility). From Table 1, it is observed that the F-score of ORG type is the lowest among all NE types for both English and Chinese. This is because many organization names are partially recognized or missed. Besides, not shown in the table, the location names or abbreviated organization names tend to be incorrectly recognized as person names. In general, the initial Chinese NER outperforms the initial English NER, as the NE type classification turns out to be a more difficult problem for this English NER system.

When those initially identified NEs are directly used for baseline alignment, only 64.1% F score (regard of their name types) is obtained. Such a low performance is mainly due to those NE recognition errors which have been brought into the alignment stage.

To diminish the effect of errors accumulating, which stems from the recognition stage, the baseline system also adopts the same expansion strategy described in Section 3.3 to enlarge the possi-

[3] The LDC2005T34 data-set consists of proofread bilingual entries: 73,352 person names, 76,460 location names and 68,960 organization names.

[4] http://mallet.cs.umass.edu/index.php/Main_Page

ble NE candidate set. However, only a slight improvement (68.4% type-sensitive F-score) is obtained, as shown in Table 2. Therefore, it is conjectured that the baseline alignment model is unable to achieve good performance if those features/factors proposed in this paper are not adopted.

4.2 The Recognition and Alignment Joint Model

To show the individual effect of each factor in the joint model, a series of experiments, from Exp0 to Exp11, are conducted. Exp0 is the *basic system*, which ignores monolingual candidate certainty scores, and also disregards mapping type and NE type consistency constraint by ignoring $P(Mtype_n \mid ew_{[n]}, RType)$ and $P(\delta \mid RType)$, and also replacing $P(cpn_{<n>} \mid Mtype_n, ew_{[n]}, RType)$ with $P(cpn_{<n>} \mid ew_{[n]})$ in Eq (3).

To show the effect of enforcing NE type consistency constraint on internal component mapping, Exp1 (named *Exp0+RType*) replaces $P(cpn_{<n>} \mid ew_{[n]})$ in Exp0 with $P(cpn_{<n>} \mid ew_{[n]}, RType)$; On the other hand, Exp2 (named *Exp0+MappingType*) shows the effect of introducing the component mapping type to Eq (3) by replacing $P(cpn_{<n>} \mid ew_{[n]})$ in Exp0 by $P(cpn_{<n>} \mid Mtype_n, ew_{[n]}) \times P(Mtype_n \mid ew_{[n]})$; Then Exp3 (named *Exp2+MappingTypeRatio*) further adds $P(\delta \mid RType)$ to Exp2, to manifest the contribution from the mapping type ratio. In addition, Exp4 (named *Exp0+RTypeReassignment*) adds the NE type reassignment score, Eq (4), to Exp0 to show the effect of enforcing NE-type consistency. Furthermore, Exp5 (named *All-BiFactors*) shows the full power of the set of proposed bilingual factors by turning on all the options mentioned above. As the bilingual alignment factors would favor the candidates with shorter lengths, $P([cpn_{<n>}, ew_{[n]}, Mtype_n]_{n=1}^{N}, \delta \mid RType, RENE)$, Eq (3), is further *normalized* into the following form:

$$\left[\prod_{n=1}^{N} P(cpn_{<n>} \mid Mtype_n, ew_{[n]}, RType) \times P(Mtype_n \mid ew_{[n]}, RType)\right]^{\frac{1}{N}} \times P(\delta \mid RType),$$

and is shown by Exp6 (named *All-N-BiFactors*).

To show the influence of additional information carried by those initially recognized NEs, Exp7 (named *Exp6+LeftD/RightD*) adds left and right distance information into Exp6, as that specified in Eq (5). To study the monolingual bigram capability, Exp8 (named *Exp6+Bigram*) adds the NEtype dependant bigram model of each language to Exp6. We use SRI Language Modeling Toolkit[5] (SRILM) (Stolcke, 2002) to train various character/word based bi-gram models with different NE types. Similar to what we have done on the bilingual alignment factor above, Exp9 (named *Exp6+N-Bigram*) adds the *normalized* NEtype dependant bigram to Exp6 for removing the bias induced by having different NE lengths. The normalized Chinese NEtype dependant bigram score is defined as $[\prod_{m=1}^{M} P(cc_m \mid cc_{m-1}, RType)]^{\frac{1}{M}}$. A Similar transformation is also applied to the English side.

Lastly, Exp10 (named *Fully-JointModel*) shows the full power of the proposed Recognition and Alignment Joint Model by adopting all the normalized factors mentioned above. The result of a MERT weighted version is further shown by Exp11 (named *Weighted-JointModel*).

Model	P (%)	R (%)	F (%)
Baseline	77.1 (67.1)	79.7 (69.8)	78.4 (68.4)
Exp0 (*Basic System*)	67.9 (62.4)	70.3 (64.8)	69.1 (63.6)
Exp1 (*Exp0 + Rtype*)	69.6 (65.7)	71.9 (68.0)	70.8 (66.8)
Exp2 (*Exp0 + MappingType*)	70.5 (65.3)	73.0 (67.5)	71.7 (66.4)
Exp3 (*Exp2 + MappingTypeRatio*)	72.0 (68.3)	74.5 (70.8)	73.2 (69.5)
Exp4 (*Exp0 + RTypeReassignment*)	70.2 (66.7)	72.7 (69.2)	71.4 (67.9)
Exp5 (*All-BiFactors*)	76.2 (**72.3**)	78.5 (**74.6**)	77.3 (**73.4**)
Exp6 (*All-N-BiFactors*)	**77.7** (**73.5**)	79.9 (**75.7**)	78.8 (**74.6**)
Exp7 (*Exp6 + LeftD/RightD*)	**83.5** (**77.7**)	**85.8** (**80.1**)	**84.6** (**78.9**)
Exp8 (*Exp6 + Bigram*)	**80.4** (**75.5**)	**82.7** (**77.9**)	**81.5** (**76.7**)
Exp9 (*Exp6 + N-Bigram*)	**82.7** (**77.1**)	**85.1** (**79.6**)	**83.9** (**78.3**)
Exp10 (*Fully-JointModel*)	**83.7** (**78.1**)	**86.2** (**80.7**)	**84.9** (**79.4**)
Exp11 (*Weighted-Joint Model*)	**85.9** (**80.5**)	**88.4** (**83.0**)	**87.1** (**81.7**)

Table 2. NEA Type-Insensitive (Type-Sensitive) Performance

Since most papers in the literature are evaluated only based on the boundaries of NEs, two kinds of performance are thus given here. The first one (named type-insensitive) only checks the scope of each NE without taking its associated NE type into consideration, and is reported

[5] http://www.speech.sri.com/projects/srilm/

as the main data at Table 2. The second one (named type-sensitive) would also evaluate the associated NE type of each NE, and is given within parentheses in Table 2. A large degradation is observed when NE type is also taken into account. The highlighted entries are those that are statistically better[6] than that of the baseline system.

4.3 ME Approach with Primitive Features

Although the proposed model has been derived above in a principled way, since all these proposed features can also be directly integrated with the well-known maximum entropy (ME) (Berger et al., 1996) framework without making any assumptions, one might wonder if it is still worth to deriving a model after all the related features have been proposed. To show that not only the features but also the adopted model contribute to the performance improvement, an ME approach is tested as follows for comparison. It directly adopts all those primitive features mentioned above as its inputs (including internal component mapping, initial and final NE type, NE bigram-based string, and left/right distance), without involving any related probability factors derived within the proposed model.

This ME method is implemented with a public package YASMET[7], and is tested under various training-set sizes (400, 4,000, 40,000, and 90,412 sentence-pairs). All those training-sets are extracted from the Training-Set-I mentioned above (a total of 298,302 NE pairs included are manually labeled). Since the ME approach is unable to utilize the bilingual NE dictionary (Training-Set-II), for fair comparison, this dictionary was also not used to train our models here. Table 3 shows the performance (F-score) using the same testing-set. The data within parentheses are relative improvements.

Model	400	4,000	40,000	90,412
ME framework	36.5 (0%)	50.4 (0%)	62.6 (0%)	67.9 (0%)
Un-weighted-JointModel	+4.6 (+12.6%)	+4.5 (+8.9%)	+4.3 (+6.9%)	+4.1 (+6.0%)
Weighted-JointModel	+5.0 (+13.7%)	+4.7 (+9.3%)	+4.6 (+7.3%)	+4.5 (+6.6%)

Table 3. Comparison between ME Framework and Derived Model on the Testing-Set

[6] Statistical significance test is measured on 95% confidence level on 1,000 re-sampling batches (Zhang et al., 2004)
[7] http://www.fjoch.com/YASMET.html

The improvement indicated in Table 3 clearly illustrates the benefit of deriving the model shown in Eq (2). Since a reasonably derived model not only shares the same training-set with the primitive ME version above, but also enjoys the additional knowledge introduced by the human (i.e., the assumptions/constraints implied by the model), it is not surprising to find out that a good model does help, and that it also becomes more noticeable as the training-set gets smaller.

5 Error Analysis and Discussion

Although the proposed model has substantially improved the performance of both NE alignment and recognition, some errors still remain. Having examined those type-insensitive errors, we found that they can be classified into four categories: (A) Original NEs or their components are already not one-to-one mapped (23%). (B) NE components are one-to-one linked, but the associated NE anchors generated from the initial recognition stage are either missing or spurious (24%). Although increasing the number of output candidates generated from the initial recognition stage might cover the missing problem, possible side effects might also be expected (as the complexity of the alignment task would also be increased). (C) Mapping types are not assumed by the model (27%). For example, one NE is abbreviated while its counterpart is not; or some loanwords or out-of-vocabulary terms are translated neither semantically nor phonetically. (D) Wrong NE scopes are selected (26%). Errors of this type are uneasy to resolve, and their possible solutions are beyond the scope of this paper.

Examples of above category (C) are interesting and are further illustrated as follows. As an instance of abbreviation errors, a Chinese NE "葛兰素制药厂 (GlaxoSmithKline Factory)" is tagged as "葛兰素/PRR 制药厂/n", while its counterpart in the English side is simply abbreviated as "GSK" (or replaced by a pronoun "it" sometimes). Linking "葛兰素" to "GSK" (or to the pronoun "it") is thus out of reach of our model. It seems an abbreviation table (or even anaphora analysis) is required to recover these kind of errors.

As an example of errors resulting from loanwords; Japanese kanji "明仁" (the name of a Japanese emperor) is linked to the English word "Akihito". Here the Japanese kanji "明仁" is directly adopted as the corresponding Chinese characters (as those characters were originally borrowed from Chinese), which would be pro-

nounced as “Mingren” in Chinese and thus deviates greatly from the English pronunciation of “Akihito”. Therefore, it is translated neither semantically nor phonetically. Further extending the model to cover this new conversion type seems necessary; however, such a kind of extension is very likely to be language pair dependent.

6 Capability of the Proposed Model

In addition to improving NE alignment, the proposed joint model can also boost the performance of NE recognition in both languages. The corresponding differences in performance (of the weighted version) when compared with the initial NER (ΔP, ΔR and ΔF) are shown in Table 4. Again, those marked entries indicate that they are statistically better than that of the original NER.

NEtype	ΔP (%): C/E	ΔR (%): C/E	ΔF (%): C/E
PER	**+5.4** / **+6.4**	+2.2 / **+2.6**	**+3.9** / **+4.6**
LOC	**+4.0** / **+3.4**	-0.2 / **+2.7**	+1.8 / **+3.0**
ORG	**+7.0** / **+3.9**	**+5.6** / **+9.1**	**+6.2** / **+6.4**
ALL	**+5.3** /**+5.2**	**+2.4** / **+4.0**	**+3.9** / **+4.6**

Table 4. Improvement in Chinese/English NER

The result shows that the proposed joint model has a clear win over the initial NER for either Chinese or English NER. In particular, ORG seems to have yielded the greatest gain amongst NE types, which matches our previous observations that the boundaries of Chinese ORG are difficult to identify with the information only coming from the Chinese sentence, while the type of English ORG is uneasy to classify with the information only coming from the English sentence.

Though not shown in the tables, it is also observed that the proposed approach achieves a 28.9% reduction on the spurious (false positive) and partial tags over the initial Chinese NER, as well as 16.1% relative error reduction compared with the initial English NER. In addition, total 27.2% wrong Chinese NEs and 40.7% wrong English NEs are corrected into right NE types. However, if the mapping type ratio is omitted, only 21.1% wrong Chinese NE types and 34.8% wrong English NE types can be corrected. This clearly indicates that the ratio is essential for identifying NE types.

With the benefits shown above, the alignment model could thus be used to train the monolingual NE recognition model via semi-supervised learning. This advantage is important for updating the NER model from time to time, as various domains frequently have different sets of NEs and new NEs also emerge with time.

Since the Chinese NE recognizer we use is not an open source toolkit, it cannot be used to carry out semi-supervised learning. Therefore, only the English NE recognizer and the alignment model are updated during training iterations. In our experiments, 50,412 sentence pairs are first extracted from Training-Set-I as unlabeled data. Various labeled data-sets are then extracted from the remaining data as different seed corpora (100, 400, 4,000 and 40,000 sentence-pairs). Table 5 shows the results of semi-supervised learning after convergence for adopting only the English NER model (*NER-Only*), the baseline alignment model (*NER+Baseline*), and our un-weighted joint model (*NER+JointModel*) respectively. The *Initial-NER* row indicates the initial performance of the NER model re-trained from different seed corpora. The data within parentheses are relative improvement over *Initial-NER*. Note that the testing set is still the same as before.

As Table 5 shows, with the NER model alone, the performance may even deteriorate after convergence. This is due to the fact that maximizing likelihood does not imply minimizing the error rate. However, with additional mapping constraints from the aligned sentence of another language, the alignment module could guide the searching process to converge to a more desirable point in the parameter space; and these additional constraints become more effective as the seed-corpus gets smaller.

Model	100	400	4,000	40,000
Initial-NER	36.7 (0%)	58.6 (0%)	71.4 (0%)	79.1 (0%)
NER-Only	-2.3 (-6.3%)	-0.5 (-0.8%)	-0.3 (-0.4%)	-0.1 (-0.1%)
NER+Baseline	+4.9 (+13.4%)	+3.4 (5.8%)	+1.7 (2.4%)	+0.7 (0.9%)
NER+Joint Model	+10.7 (+29.2%)	+8.7 (+14.8%)	+4.8 (+6.7%)	+2.3 (+2.9%)

Table 5. Testing-Set Performance for Semi-Supervised Learning of English NE Recognition

7 Conclusion

In summary, our experiments show that the new monolingual candidate certainty factors are more effective than the tagging cost (only bigram model) adopted in the baseline system. Moreover, both the mapping type ratio and the entity type consistency constraint are very helpful in identifying the associated NE boundaries and types. After having adopted the features and enforced

the constraint mentioned above, the proposed framework, which jointly recognizes and aligns bilingual named entities, achieves a remarkable 42.1% imperfection reduction on type-sensitive F-score (from 68.4% to 81.7%) in our Chinese-English NE alignment task.

Although the experiments are conducted on the Chinese-English language pair, it is expected that the proposed approach can also be applied to other language pairs, as no language dependent linguistic feature (or knowledge) is adopted in the model/algorithm used.

Acknowledgments

The research work has been partially supported by the National Natural Science Foundation of China under Grants No. 60975053, 90820303, and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, and also the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4.

References

Al-Onaizan, Yaser, and Kevin Knight. 2002. Translating Named Entities Using Monolingual and Bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 400-408.

Berger, Adam L., Stephen A. Della Pietra and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39-72, March.

Chen, Hsin-His, Changhua Yang and Ying Lin. 2003. Learning Formulation and Transformation Rules for Multilingual Named Entities. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, pages 1-8.

Feng, Donghui, Yajuan Lv and Ming Zhou. 2004. A New Approach for English-Chinese Named Entity Alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 372-379.

Huang, Fei, Stephan Vogel and Alex Waibel. 2003. Automatic Extraction of Named Entity Translingual Equivalence Based on Multi-Feature Cost Minimization. In *Proceedings of ACL'03, Workshop on Multilingual and Mixed-language Named Entity Recognition*. Sappora, Japan.

Ji, Heng and Ralph Grishman. 2006. Analysis and Repair of Name Tagger Errors. In *Proceedings of COLING/ACL 06*, Sydney, Australia.

Lee, Chun-Jen, Jason S. Chang and Jyh-Shing R. Jang. 2006. Alignment of Bilingual Named Entities in Parallel Corpora Using Statistical Models and Multiple Knowledge Sources. *ACM Transactions on Asian Language Information Processing (TALIP)*, 5(2): 121-145.

Moore, R. C.. 2003. Learning Translations of Named-Entity Phrases from Parallel Corpora. In *Proceedings of 10th Conference of the European Chapter of ACL*, Budapest, Hungary.

Och, Franz Josef. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of the 41st Annual Conference of the Association for Computational Linguistics (ACL). July 8-10, 2003. Sapporo, Japan. Pages: 160-167.

Stolcke, A. 2002. SRILM -- An Extensible Language Modeling Toolkit. *Proc. Intl. Conf. on Spoken Language Processing*, vol. 2, pp. 901-904, Denver.

Wu, Youzheng, Jun Zhao and Bo Xu. 2005. Chinese Named Entity Recognition Model Based on Multiple Features. In *Proceedings of HLT/EMNLP* 2005, pages 427-434.

Zhang, Ying, Stephan Vogel, and Alex Waibel, 2004. Interpreting BLEU/NIST Scores: How Much Improvement Do We Need to Have a Better System? In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 2051--2054.

Generating Templates of Entity Summaries with an Entity-Aspect Model and Pattern Mining

Peng Li[1] and **Jing Jiang**[2] and **Yinglin Wang**[1]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]School of Information Systems, Singapore Management University
{lipeng,ylwang}@sjtu.edu.cn jingjiang@smu.edu.sg

Abstract

In this paper, we propose a novel approach to automatic generation of summary templates from given collections of summary articles. This kind of summary templates can be useful in various applications. We first develop an entity-aspect LDA model to simultaneously cluster both sentences and words into aspects. We then apply frequent subtree pattern mining on the dependency parse trees of the clustered and labeled sentences to discover sentence patterns that well represent the aspects. Key features of our method include automatic grouping of semantically related sentence patterns and automatic identification of template slots that need to be filled in. We apply our method on five Wikipedia entity categories and compare our method with two baseline methods. Both quantitative evaluation based on human judgment and qualitative comparison demonstrate the effectiveness and advantages of our method.

1 Introduction

In this paper, we study the task of automatically generating templates for entity summaries. An entity summary is a short document that gives the most important facts about an entity. In Wikipedia, for instance, most articles have an introduction section that summarizes the subject entity before the table of contents and other elaborate sections. These introduction sections are examples of entity summaries we consider. Summaries of entities from the same category usually share some common structure. For example, biographies of physicists usually contain facts about the nationality, educational background, affiliation and major contributions of the physicist, whereas introductions of companies usually list information such as the industry, founder and headquarter of the company. Our goal is to automatically construct a summary template that outlines the most salient types of facts for an entity category, given a collection of entity summaries from this category.

Such kind of summary templates can be very useful in many applications. First of all, they can uncover the underlying structures of summary articles and help better organize the information units, much in the same way as infoboxes do in Wikipedia. In fact, automatic template generation provides a solution to induction of infobox structures, which are still highly incomplete in Wikipedia (Wu and Weld, 2007). A template can also serve as a starting point for human editors to create new summary articles. Furthermore, with summary templates, we can potentially apply information retrieval and extraction techniques to construct summaries for new entities automatically on the fly, improving the user experience for search engine and question answering systems.

Despite its usefulness, the problem has not been well studied. The most relevant work is by Filatova et al. (2006) on automatic creation of domain templates, where the defintion of a domain is similar to our notion of an entity category. Filatova et al. (2006) first identify the important verbs for a domain using corpus statistics, and then find frequent parse tree patterns from sentences containing these verbs to construct a domain template. There are two major limitations of their approach. First, the focus on verbs restricts the template patterns that can be found. Second, redundant or related patterns using different verbs to express the same or similar facts cannot be grouped together. For example, "*won X award*" and "*received X prize*" are considered two different patterns by this approach. We propose a method that can overcome these two limitations. Automatic template generation is also related to a number of other problems that have been studied before, in-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 640–649,
Uppsala, Sweden, 11-16 July 2010.

cluding unsupervised IE pattern discovery (Sudo et al., 2003; Shinyama and Sekine, 2006; Sekine, 2006; Yan et al., 2009) and automatic generation of Wikipedia articles (Sauper and Barzilay, 2009). We discuss the differences of our work from existing related work in Section 6.

In this paper we propose a novel approach to the task of automatically generating entity summary templates. We first develop an entity-aspect model that extends standard LDA to identify clusters of words that can represent different aspects of facts that are salient in a given summary collection (Section 3). For example, the words "received," "award," "won" and "Nobel" may be clustered together from biographies of physicists to represent one aspect, even though they may appear in different sentences from different biographies. Simultaneously, the entity-aspect model separates words in each sentence into background words, document words and aspect words, and sentences likely about the same aspect are naturally clustered together. After this aspect identification step, we mine frequent subtree patterns from the dependency parse trees of the clustered sentences (Section 4). Different from previous work, we leverage the word labels assigned by the entity-aspect model to prune the patterns and to locate template slots to be filled in.

We evaluate our method on five entity categories using Wikipedia articles (Section 5). Because the task is new and thus there is no standard evaluation criteria, we conduct both quantitative evaluation using our own human judgment and qualitative comparison. Our evaluation shows that our method can obtain better sentence patterns in terms of f1 measure compared with two baseline methods, and it can also achieve reasonably good quality of aspect clusters in terms of purity. Compared with standard LDA and K-means sentence clustering, the aspects identified by our method are also more meaningful.

2 The Task

Given a collection of entity summaries from the same entity category, our task is to automatically construct a summary template that outlines the most important information one should include in a summary for this entity category. For example, given a collection of biographies of physicists, ideally the summary template should indicate that important facts about a physicist include his/her educational background, affiliation, major contributions, awards received, etc.

Aspect	Pattern
1	*ENT* received his phd from ? university *ENT* studied ? under ? *ENT* earned his ? in physics from university of ?
2	*ENT* was awarded the medal in ? *ENT* won the ? award *ENT* received the nobel prize in physics in ?
3	*ENT* was ? director *ENT* was the head of ? *ENT* worked for ?
4	*ENT* made contributions to ? *ENT* is best known for work on ? *ENT* is noted for ?

Table 1: Examples of some good template patterns and their aspects generated by our method.

However, it is not clear what is the best representation of such templates. Should a template comprise a list of subtopic labels (e.g. "education" and "affiliation") or a set of explicit questions? Here we define a template format based on the usage of the templates as well as our observations from Wikipedia entity summaries. First, since we expect that the templates can be used by human editors for creating new summaries, we use sentence patterns that are human readable as basic units of the templates. For example, we may have a sentence pattern "*ENT* graduated from ? University" for the entity category "physicist," where *ENT* is a placeholder for the entity that the summary is about, and '?' is a slot to be filled in. Second, we observe that information about entities of the same category can be grouped into subtopics. For example, the sentences "Bohr is a Nobel laureate" and "Einstein received the Nobel Prize" are paraphrases of the same type of facts, while the sentences "Taub earned his doctorate at Princeton University" and "he graduated from MIT" are slightly different but both describe a person's educational background. Therefore, it makes sense to group sentence patterns based on the subtopics they pertain to. Here we call these subtopics the *aspects* of a summary template.

Formally, we define a summary template to be a set of sentence patterns grouped into aspects. Each sentence pattern has a placeholder for the entity to be summarized and possibly one or more template slots to be filled in. Table 1 shows some sentence patterns our method has generated for the "physicist" category.

2.1 Overview of Our Method

Our automatic template generation method consists of two steps:

Aspect Identification: In this step, our goal is to automatically identify the different aspects or subtopics of the given summary collection. We simultaneously cluster sentences and words into aspects, using an entity-aspect model extended from the standard LDA model that is widely used in text mining (Blei et al., 2003). The output of this step are sentences clustered into aspects, with each word labeled as a stop word, a background word, a document word or an aspect word.

Sentence Pattern Generation: In this step, we generate human-readable sentence patterns to represent each aspect. We use frequent subtree pattern mining to find the most representative sentence structures for each aspect. The fixed structure of a sentence pattern consists of aspect words, background words and stop words, while document words become template slots whose values can vary from summary to summary.

3 Aspect Identification

At the aspect identification step, our goal is to discover the most salient aspects or subtopics contained in a summary collection. Here we propose a principled method based on a modified LDA model to simultaneously cluster both sentences and words to discover aspects.

We first make the following observation. In entity summaries such as the introduction sections of Wikipedia articles, most sentences are talking about a single fact of the entity. If we look closely, there are a few different kinds of words in these sentences. First of all, there are stop words that occur frequently in any document collection. Second, for a given entity category, some words are generally used in all aspects of the collection. Third, some words are clearly associated with the aspects of the sentences they occur in. And finally, there are also words that are document or entity specific. For example, in Table 2 we show two sentences related to the "affiliation" aspect from the "physicist" summary collection. Stop words such as "is" and "the" are labeled with "S." The word "physics" can be regarded as a background word for this collection. "Professor" and "university" are clearly related to the "affiliation" aspect. Finally words such as "Modena" and "Chicago" are specifically associated with the subject entities being discussed, that is, they are specific to the summary documents.

To capture background words and document-specific words, Chemudugunta et al. (2007) proposed to introduce a background topic and document-specific topics. Here we borrow their idea and also include a background topic as well as document-specific topics. To discover aspects that are local to one or a few adjacent sentences but may occur in many documents, Titov and McDonald (2008) proposed a multi-grain topic model, which relies on word co-occurrences within short paragraphs rather than documents in order to discover aspects. Inspired by their model, we rely on word co-occurrences within single sentences to identify aspects.

3.1 Entity-Aspect Model

We now formally present our entity-aspect model. First, we assume that stop words can be identified using a standard stop word list. We then assume that for a given entity category there are three kinds of unigram language models (i.e. multinomial word distributions). There is a background model $\phi^{\mathcal{B}}$ that generates words commonly used in all documents and all aspects. There are D document models ψ^d ($1 \leq d \leq D$), where D is the number of documents in the given summary collection, and there are A aspect models ϕ^a ($1 \leq a \leq A$), where A is the number of aspects. We assume that these word distributions have a uniform Dirichlet prior with parameter β.

Since not all aspects are discussed equally frequently, we assume that there is a global aspect distribution θ that controls how often each aspect occurs in the collection. θ is sampled from another Dirichlet prior with parameter α. There is also a multinomial distribution π that controls in each sentence how often we encounter a background word, a document word, or an aspect word. π has a Dirichlet prior with parameter γ.

Let S_d denote the number of sentences in document d, $N_{d,s}$ denote the number of words (after stop word removal) in sentence s of document d, and $w_{d,s,n}$ denote the n'th word in this sentence. We introduce hidden variables $z_{d,s}$ for each sentence to indicate the aspect a sentence belongs to. We also introduce hidden variables $y_{d,s,n}$ for each word to indicate whether a word is generated from the background model, the document model, or the aspect model. Figure 1 shows the process of

```
    Venturi/D is/S a/S professor/A of/S physics/B at/S the/S University/A of/S
Modena/D ./S
    He/S was/S a/S professor/A of/S physics/B at/S the/S University/A of/S
Chicago/D until/S 1982/D ./S
```

Table 2: Two sentences on "affiliation" from the "physicist" entity category. S: stop word. B: background word. A: aspect word. D: document word.

1. Draw $\theta \sim \text{Dir}(\alpha), \phi^{\mathcal{B}} \sim \text{Dir}(\beta), \pi \sim \text{Dir}(\gamma)$
2. For each aspect $a = 1, \ldots, A$,
   (a) draw $\phi^a \sim \text{Dir}(\beta)$
3. For each document $d = 1, \ldots, D$,
   (a) draw $\psi^d \sim \text{Dir}(\beta)$
   (b) for each sentence $s = 1, \ldots, S_d$
      i. draw $z_{d,s} \sim \text{Multi}(\theta)$
      ii. for each word $n = 1, \ldots, N_{d,s}$
         A. draw $y_{d,s,n} \sim \text{Multi}(\pi)$
         B. draw $w_{d,s,n} \sim \text{Multi}(\phi^{\mathcal{B}})$ if $y_{d,s,n} = 1$, $w_{d,s,n} \sim \text{Multi}(\psi^d)$ if $y_{d,s,n} = 2$, or $w_{d,s,n} \sim \text{Multi}(\phi^{z_{d,s}})$ if $y_{d,s,n} = 3$

Figure 1: The document generation process.

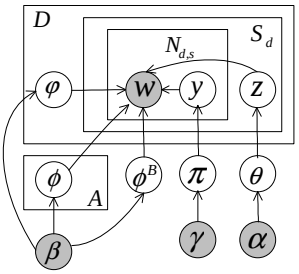


Figure 2: The entity-aspect model.

generating the whole document collection. The plate notation of the model is shown in Figure 2. Note that the values of α, β and γ are fixed. The number of aspects A is also manually set.

3.2 Inference

Given a summary collection, i.e. the set of all $w_{d,s,n}$, our goal is to find the most likely assignment of $z_{d,s}$ and $y_{d,s,n}$, that is, the assignment that maximizes $p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{w}; \alpha, \beta, \gamma)$, where $\boldsymbol{z}$, $\boldsymbol{y}$ and $\boldsymbol{w}$ represent the set of all z, y and w variables, respectively. With the assignment, sentences are naturally clustered into aspects, and words are labeled as either a background word, a document word, or an aspect word.

We approximate $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{w}; \alpha, \beta, \gamma)$ by $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{w}; \hat{\phi}^{\mathcal{B}}, \{\hat{\psi}^d\}_{d=1}^D, \{\hat{\phi}^a\}_{a=1}^A, \hat{\theta}, \hat{\pi})$, where $\hat{\phi}^{\mathcal{B}}$, $\{\hat{\psi}^d\}_{d=1}^D$, $\{\hat{\phi}^a\}_{a=1}^A$, $\hat{\theta}$ and $\hat{\pi}$ are estimated using Gibbs sampling, which is commonly used for inference for LDA models (Griffiths and Steyvers, 2004). Due to space limit, we give the formulas for the Gibbs sampler below without derivation.

First, given sentence s in document d, we sample a value for $z_{d,s}$ given the values of all other z and y variables using the following formula:

$$p(z_{d,s} = a|\boldsymbol{z}_{\neg\{d,s\}}, \boldsymbol{y}, \boldsymbol{w}) \propto \frac{C^{\mathcal{A}}_{(a)} + \alpha}{C^{\mathcal{A}}_{(\cdot)} + A\alpha} \cdot \frac{\prod_{v=1}^{V} \prod_{i=0}^{E_{(v)}} (C^a_{(v)} + i + \beta)}{\prod_{i=0}^{E_{(\cdot)}} (C^a_{(\cdot)} + i + V\beta)}.$$

In the formula above, $\boldsymbol{z}_{\neg\{d,s\}}$ is the current aspect assignment of all sentences excluding the current sentence. $C^{\mathcal{A}}_{(a)}$ is the number of sentences assigned to aspect a, and $C^{\mathcal{A}}_{(\cdot)}$ is the total number of sentences. V is the vocabulary size. $C^a_{(v)}$ is the number of times word v has been assigned to aspect a. $C^a_{(\cdot)}$ is the total number of words assigned to aspect a. All the counts above exclude the current sentence. $E_{(v)}$ is the number of times word v occurs in the current sentence and is assigned to be an aspect word, as indicated by $\boldsymbol{y}$, and $E_{(\cdot)}$ is the total number of words in the current sentence that are assigned to be an aspect word.

We then sample a value for $y_{d,s,n}$ for each word in the current sentence using the following formulas:

$$p(y_{d,s,n} = 1|\boldsymbol{z}, \boldsymbol{y}_{\neg\{d,s,n\}}) \propto \frac{C^{\pi}_{(1)} + \gamma}{C^{\pi}_{(\cdot)} + 3\gamma} \cdot \frac{C^{\mathcal{B}}_{(w_{d,s,n})} + \beta}{C^{\mathcal{B}}_{(\cdot)} + V\beta},$$

$$p(y_{d,s,n} = 2|\boldsymbol{z}, \boldsymbol{y}_{\neg\{d,s,n\}}) \propto \frac{C^{\pi}_{(2)} + \gamma}{C^{\pi}_{(\cdot)} + 3\gamma} \cdot \frac{C^{d}_{(w_{d,s,n})} + \beta}{C^{d}_{(\cdot)} + V\beta},$$

$$p(y_{d,s,n} = 3|\boldsymbol{z}, \boldsymbol{y}_{\neg\{d,s,n\}}) \propto \frac{C^{\pi}_{(3)} + \gamma}{C^{\pi}_{(\cdot)} + 3\gamma} \cdot \frac{C^{a}_{(w_{d,s,n})} + \beta}{C^{a}_{(\cdot)} + V\beta}.$$

In the formulas above, $\boldsymbol{y}_{\neg\{d,s,n\}}$ is the set of all y variables excluding $y_{d,s,n}$. $C^{\pi}_{(1)}$, $C^{\pi}_{(2)}$ and $C^{\pi}_{(3)}$ are the numbers of words assigned to be a background word, a document word, or an aspect word, respectively, and $C^{\pi}_{(\cdot)}$ is the total number of words. $C^{\mathcal{B}}$ and C^d are counters similar to C^a but are for the background model and the document models. In all these counts, the current word is excluded.

With one Gibbs sample, we can make the following estimation:

$$\hat{\phi}_v^B = \frac{C^{\mathcal{B}}_{(v)} + \beta}{C^{\mathcal{B}}_{(\cdot)} + V\beta}, \hat{\psi}_v^d = \frac{C^d_{(v)} + \beta}{C^d_{(\cdot)} + V\beta}, \hat{\phi}_v^a = \frac{C^a_{(v)} + \beta}{C^a_{(\cdot)} + V\beta},$$

$$\hat{\theta}_a = \frac{C^{\mathcal{A}}_{(a)} + \alpha}{C^{\mathcal{A}}_{(\cdot)} + A\alpha}, \hat{\pi}_t = \frac{C^{\pi}_{(t)} + \gamma}{C^{\pi}_{(\cdot)} + 3\gamma} (1 \leq t \leq 3).$$

Here the counts include all sentences and all words.

In our experiments, we set $\alpha = 5$, $\beta = 0.01$ and $\gamma = 20$. We run 100 burn-in iterations through all documents in a collection to stabilize the distribution of $\boldsymbol{z}$ and $\boldsymbol{y}$ before collecting samples. We found that empirically 100 burn-in iterations were sufficient for our data set. We take 10 samples with a gap of 10 iterations between two samples, and average over these 10 samples to get the estimation for the parameters.

After estimating $\hat{\phi}^{\mathcal{B}}$, $\{\hat{\psi}^d\}_{d=1}^D$, $\{\hat{\phi}^a\}_{a=1}^A$, $\hat{\theta}$ and $\hat{\pi}$, we find the values of each $z_{d,s}$ and $y_{d,s,n}$ that maximize $p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{w}; \hat{\phi}^{\mathcal{B}}, \{\hat{\psi}^d\}_{d=1}^D, \{\hat{\phi}^a\}_{a=1}^A, \hat{\theta}, \hat{\pi})$. This assignment, together with the standard stop word list we use, gives us sentences clustered into A aspects, where each word is labeled as either a stop word, a background word, a document word or an aspect word.

3.3 Comparison with Other Models

A major difference of our entity-aspect model from standard LDA model is that we assume each sentence belongs to a single aspect while in LDA words in the same sentence can be assigned to different topics. Our one-aspect-per-sentence assumption is important because our goal is to cluster sentences into aspects so that we can mine common sentence patterns for each aspect.

To cluster sentences, we could have used a straightforward solution similar to document clustering, where sentences are represented as feature vectors using the vector space model, and a standard clustering algorithm such as K-means can be applied to group sentences together. However, there are some potential problems with directly applying this typical document clustering method. First, unlike documents, sentences are short, and the number of words in a sentence that imply its aspect is even smaller. Besides, we do not know the aspect-related words in advance. As a result, the cosine similarity between two sentences may not reflect whether they are about the same aspect. We can perform heuristic term weighting, but the method becomes less robust. Second, after sentence clustering, we may still want to identify the the aspect words in each sentence, which are useful in the next pattern mining step. Directly taking the most frequent words from each sentence cluster as aspect words may not work well even after stop word removal, because there can be background words commonly used in all aspects.

4 Sentence Pattern Generation

At the pattern generation step, we want to identify human-readable sentence patterns that best represent each cluster. Following the basic idea from (Filatova et al., 2006), we start with the parse trees of sentences in each cluster, and apply a frequent subtree pattern mining algorithm to find sentence structures that have occurred at least K times in the cluster. Here we use dependency parse trees.

However, different from (Filatova et al., 2006), the word labels (S, B, D and A) assigned by the entity-aspect model give us some advantages. Intuitively, a representative sentence pattern for an aspect should contain at least one aspect word. On the other hand, document words are entity-specific and therefore should not appear in the generic template patterns; instead, they correspond to template slots that need to be filled in. Furthermore, since we work on entity summaries, in each sentence there is usually a word or phrase that refers to the subject entity, and we should have a placeholder for the subject entity in each pattern.

Based on the intuitions above, we have the following sentence pattern generation process.

1. ***Locate subject entities***: In each sentence, we want to locate the word or phrase that refers to the subject entity. For example, in a biography, usually a pronoun "he" or "she" is used to refer to the subject person. We use the following heuristic to locate the subject entities: For each summary document, we first find the top 3 frequent base noun phrases that are subjects of sentences. For example, in a company introduction, the phrase "the company" is probably used frequently as a sentence subject. Then for each sentence, we first look for the title of the Wikipedia article. If it occurs, it is tagged as the subject entity. Otherwise, we check whether one of the top 3 subject base noun phrases occurs, and if so, it is tagged as the subject entity. Otherwise, we tag the subject of the sentence as the subject entity. Finally, for the identified subject entity word or phrase, we replace the label assigned by the entity-aspect model with a

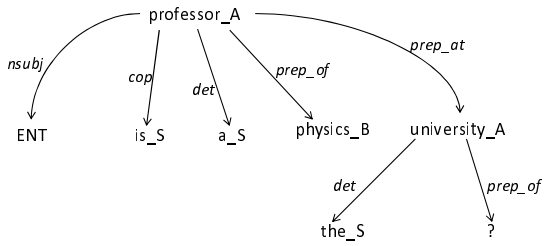


Figure 3: An example labeled dependency parse tree.

new label E.

2. ***Generate labeled parse trees***: We parse each sentence using the Stanford Parser[1]. After parsing, for each sentence we obtain a dependency parse tree where each node is a single word and each edge is labeled with a dependency relation. Each word is also labeled with one of $\{E, S, B, D, A\}$. We replace words labeled with E by a placeholder *ENT*, and replace words labeled with D by a question mark to indicate that these correspond to template slots. For the other words, we attach their labels to the tree nodes. Figure 3 shows an example labeled dependency parse tree.

3. ***Mine frequent subtree patterns***: For the set of parse trees in each cluster, we use FREQT[2], a software that implements the frequent subtree pattern mining algorithm proposed in (Zaki, 2002), to find all subtrees with a minimum support of K.

4. ***Prune patterns***: We remove subtree patterns found by FREQT that do not contain *ENT* or any aspect word. We also remove small patterns that are contained in some other larger pattern in the same cluster.

5. ***Covert subtree patterns to sentence patterns***: The remaining patterns are still represented as subtrees. To covert them back to human-readable sentence patterns, we map each pattern back to one of the sentences that contain the pattern to order the tree nodes according to their original order in the sentence.

In the end, for each summary collection, we obtain A clusters of sentence patterns, where each cluster presumably corresponds to a single aspect or subtopic.

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] http://chasen.org/~taku/software/freqt/

Category	D	S	S_d		
			min	max	avg
US Actress	407	1721	1	21	4
Physicist	697	4238	1	49	6
US CEO	179	1040	1	24	5
US Company	375	2477	1	36	6
Restaurant	152	1195	1	37	7

Table 3: The number of documents (D), total number of sentences (S) and minimum, maximum and average numbers of sentences per document (S_d) of the data set.

5 Evaluation

Because we study a non-standard task, there is no existing annotated data set. We therefore created a small data set and made our own human judgment for quantitative evaluation purpose.

5.1 Data

We downloaded five collections of Wikipedia articles from different entity categories. We took only the introduction sections of each article (before the tables of contents) as entity summaries. Some statistics of the data set are given in Table 3.

5.2 Quantitative Evaluation

To quantitatively evaluate the summary templates, we want to check (1) whether our sentence patterns are meaningful and can represent the corresponding entity categories well, and (2) whether semantically related sentence patterns are grouped into the same aspect. It is hard to evaluate both together. We therefore separate these two criteria.

5.2.1 Quality of sentence patterns

To judge the quality of sentence patterns without looking at aspect clusters, ideally we want to compute the precision and recall of our patterns, that is, the percentage of our sentence patterns that are meaningful, and the percentage of true meaningful sentence patterns of each category that our method can capture. The former is relatively easy to obtain because we can ask humans to judge the quality of our patterns. The latter is much harder to compute because we need human judges to find the set of true sentence patterns for each entity category, which can be very subjective.

We adopt the following pooling strategy borrowed from information retrieval. Assume we want to compare a number of methods that each can generate a set of sentence patterns from a summary collection. We take the union of these sets

of patterns generated by the different methods and order them randomly. We then ask a human judge to decide whether each sentence pattern is meaningful for the given category. We can then treat the set of meaningful sentence patterns found by the human judge this way as the ground truth, and precision and recall of each method can be computed. If our goal is only to compare the different methods, this pooling strategy should suffice.

We compare our method with the following two baseline methods.

Baseline 1: In this baseline, we use the same subtree pattern mining algorithm to find sentence patterns from each summary collection. We also locate the subject entities and replace them with *ENT*. However, we do not have aspect words or document words in this case. Therefore we do not prune any pattern except to merge small patterns with the large ones that contain them. The patterns generated by this method do not have template slots.

Baseline 2: In the second baseline, we apply a verb-based pruning on the patterns generated by the first baseline, similar to (Filatova et al., 2006). We first find the top-20 verbs using the scoring function below that is taken from (Filatova et al., 2006), and then prune patterns that do not contain any of the top-20 verbs.

$$s(v_i) = \frac{N(v_i)}{\sum_{v_j \in \mathcal{V}} N(v_j)} \cdot \frac{M(v_i)}{D},$$

where $N(v_i)$ is the frequency of verb v_i in the collection, $\mathcal{V}$ is the set of all verbs, D is the total number of documents in the collection, and $M(v_i)$ is the number of documents in the collection that contains v_i.

In Table 4, we show the precision, recall and f1 of the sentence patterns generated by our method and the two baseline methods for the five categories. For our method, we set the support of the subtree patterns K to 2, that is, each pattern has occurred in at least two sentences in the corresponding aspect cluster. For the two baseline methods, because sentences are not clustered, we use a larger support K of 3; otherwise, we find that there can be too many patterns. We can see that overall our method gives better f1 measures than the two baseline methods for most categories. Our method achieves a good balance between precision and recall. For BL-1, the precision is high but recall is low. Intuitively BL-1 should have a higher recall than our method because our method does more pattern pruning than BL-1 using aspect words. Here it is not the case mainly because we used a higher frequency threshold (K = 3) to select frequent patterns in BL-1, giving overall fewer patterns than in our method. For BL-2, the precision is higher than BL-1 but recall is lower. It is expected because the patterns of BL-2 is a subset of that of BL-1.

Category	B	Purity
US Actress	4	0.626
Physicist	6	0.714
US CEO	4	0.674
US Company	4	0.614
Restaurant	3	0.587

Table 5: The true numbers of aspects as judged by the human annotator (B), and the purity of the clusters.

There are some advantages of our method that are not reflected in Table 4. First, many of our patterns contain template slots, which make the pattern more meaningful. In contrast the baseline patterns do not contain template slots. Because the human judge did not give preference over patterns with slots, both "*ENT* won the award" and "*ENT* won the ? award" were judged to be meaningful without any distinction, although the former one generated by our method is more meaningful. Second, compared with BL-2, our method can obtain patterns that do not contain a non-auxiliary verb, such as "*ENT* was ? director."

5.2.2 Quality of aspect clusters

We also want to judge the quality of the aspect clusters. To do so, we ask the human judge to group the ground truth sentence patterns of each category based on semantic relatedness. We then compute the purity of the automatically generated clusters against the human judged clusters using purity. The results are shown in Table 5. In our experiments, we set the number of clusters A used in the entity-aspect model to be 10. We can see from Table 5 that our generated aspect clusters can achieve reasonably good performance.

5.3 Qualitative evaluation

We also conducted qualitative comparison between our entity-aspect model and standard LDA model as well as a K-means sentence clustering method. In Table 6, we show the top 5 frequent words of three sample aspects as found by our method, standard LDA, and K-means. Note that although we try to align the aspects, there is

Method		Category				
		US Actress	Physicist	US CEO	US Company	Restaurant
BL-1	precision	0.714	0.695	0.778	0.622	0.706
	recall	0.545	0.300	0.367	0.425	0.361
	f1	0.618	0.419	0.499	0.505	0.478
BL-2	precision	0.845	0.767	0.829	0.809	1.000
	recall	0.260	0.096	0.127	0.167	0.188
	f1	0.397	0.17	0.220	0.276	0.316
Ours	precision	0.544	0.607	0.586	0.450	0.560
	recall	0.710	0.785	0.712	0.618	0.701
	f1	0.616	0.684	0.643	0.520	0.624

Table 4: Quality of sentence patterns in terms of precision, recall and f1.

Method	Sample Aspects		
	1	2	3
Our entity-aspect model	university received ph.d. college degree	prize nobel physics awarded medal	academy sciences member national society
Standard LDA	physics american professor received university	nobel prize physicist awarded john	physics institute research member sciences
K-means	physics university institute work research	physicist american physics university nobel	physics academy sciences university new

Table 6: Comparison of the top 5 words of three sample aspects using different methods.

no correspondence between clusters numbered the same but generated by different methods.

We can see that our method gives very meaningful aspect clusters. Standard LDA also gives meaningful words, but background words such as "physics" and "physicist" are mixed with aspect words. Entity-specific words such as "john" also appear mixed with aspect words. K-means clusters are much less meaningful, with too many background words mixed with aspect words.

6 Related Work

The most related existing work is on domain template generation by Filatova et al. (2006). There are several differences between our work and theirs. First, their template patterns must contain a non-auxiliary verb whereas ours do not have this restriction. Second, their verb-centered patterns are independent of each other, whereas we group semantically related patterns into aspects, giving more meaningful templates. Third, in their work, named entities, numbers and general nouns are treated as template slots. In our method, we apply the entity-aspect model to automatically identify words that are document-specific, and treat these words as template slots, which can be potentially more robust as we do not rely on the quality of named entity recognition. Last but not least, their documents are event-centered while ours are entity-centered. Therefore we can use heuristics to anchor our patterns on the subject entities.

Sauper and Barzilay (2009) proposed a framework to learn to automatically generate Wikipedia articles. There is a fundamental difference between their task and ours. The articles they generate are long, comprehensive documents consisting of several sections on different subtopics of the subject entity, and they focus on learning the topical structures from complete Wikipedia articles. We focus on learning sentence patterns of the short, concise introduction sections of Wikipedia articles.

Our entity-aspect model is related to a number of previous extensions of LDA models. Chemudugunta et al. (2007) proposed to introduce a background topic and document-specific topics. Our background and document language models are similar to theirs. However, they still treat documents as bags of words rather than sets of sentences as in our model. Titov and McDonald (2008) exploited the idea that a short paragraph within a document is likely to be about the same aspect. Our one-aspect-per-sentence assumption is a stricter than theirs, but it is required in our model for the purpose of mining sentence patterns. The way we separate words into stop words, background words, document words and aspect words bears similarity to that used in (Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009), but their task is multi-document summarization while ours is to induce summary templates.

7 Conclusions and Future Work

In this paper, we studied the task of automatically generating templates for entity summaries. We proposed an entity-aspect model that can automatically cluster sentences and words into aspects. The model also labels words in sentences as either a stop word, a background word, a document word or an aspect word. We then applied frequent subtree pattern mining to generate sentence patterns that can represent the aspects. We took advantage of the labels generated by the entity-aspect model to prune patterns and to locate template slots. We conducted both quantitative and qualitative evaluation using five collections of Wikipedia entity summaries. We found that our method gave overall better template patterns than two baseline methods, and the aspect clusters generated by our method are reasonably good.

There are a number of directions we plan to pursue in the future in order to improve our method. First, we can possibly apply linguistic knowledge to improve the quality of sentence patterns. Currently the method may generate similar sentence patterns that differ only slightly, e.g. change of a preposition. Also, the sentence patterns may not form complete, meaningful sentences. For example, a sentence pattern may contain an adjective but not the noun it modifies. We plan to study how to use linguistic knowledge to guide the construction of sentence patterns and make them more meaningful. Second, we have not quantitatively evaluated the quality of the template slots, because our judgment is only at the whole sentence pattern level. We plan to get more human judges and more rigorously judge the relevance and usefulness of both the sentence patterns and the template slots. It is also possible to introduce certain rules or constraints to selectively form template slots rather than treating all words labeled with D as template slots.

Acknowledgments

This work was done during Peng Li's visit to the Singapore Management University. This work was partially supported by the National High-tech Research and Development Project of China (863) under the grant number 2009AA04Z106 and the National Science Foundation of China (NSFC) under the grant number 60773088. We thank the anonymous reviewers for their helpful comments.

References

David Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2007. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in Neural Information Processing Systems 19*, pages 241–248.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 305–312.

Elena Filatova, Vasileios Hatzivassiloglou, and Kathleen McKeown. 2006. Automatic creation of domain templates. In *Proceedings of 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 207–214.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl. 1):5228–5235.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Christina Sauper and Regina Barzilay. 2009. Automatically generating Wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 731–738.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 224–231.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In

Proceeding of the 17th International Conference on World Wide Web, pages 111–120.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying Wikipedia. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 41–50.

Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining Wikipedia texts using information from the Web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1021–1029.

Mohammed J. Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80.

Comparable Entity Mining from Comparative Questions

Shasha Li[1] , Chin-Yew Lin[2] , Young-In Song[2] , Zhoujun Li[3]
[1]National University of Defense Technology, Changsha, China
[2]Microsoft Research Asia, Beijing, China
[3]Beihang University, Beijing, China
shashali@nudt.edu.cn[1], {cyl,yosong}@microsoft.com[2],
lizj@buaa.edu.cn[3]

Abstract

Comparing one thing with another is a typical part of human decision making process. However, it is not always easy to know what to compare and what are the alternatives. To address this difficulty, we present a novel way to automatically mine comparable entities from comparative questions that users posted online. To ensure high precision and high recall, we develop a weakly-supervised bootstrapping method for comparative question identification and comparable entity extraction by leveraging a large online question archive. The experimental results show our method achieves F1-measure of 82.5% in comparative question identification and 83.3% in comparable entity extraction. Both significantly outperform an existing state-of-the-art method.

1 Introduction

Comparing alternative options is one essential step in decision-making that we carry out every day. For example, if someone is interested in certain products such as digital cameras, he or she would want to know what the alternatives are and compare different cameras before making a purchase. This type of comparison activity is very common in our daily life but requires high knowledge skill. Magazines such as *Consumer Reports* and *PC Magazine* and online media such as *CNet.com* strive in providing editorial comparison content and surveys to satisfy this need.

In the World Wide Web era, a comparison activity typically involves: search for relevant web pages containing information about the targeted products, find competing products, read reviews, and identify pros and cons. In this paper, we focus on finding a set of comparable entities given a user's input entity. For example, given an entity, *Nokia N95* (a cellphone), we want to find comparable entities such as *Nokia N82*, *iPhone* and so on.

In general, it is difficult to decide if two entities are comparable or not since people do compare apples and oranges for various reasons. For example, "*Ford*" and "*BMW*" might be comparable as "car manufacturers" or as "market segments that their products are targeting", but we rarely see people comparing "*Ford Focus*" (car model) and "*BMW 328i*". Things also get more complicated when an entity has several functionalities. For example, one might compare "*iPhone*" and "*PSP*" as "portable game player" while compare "*iPhone*" and "*Nokia N95*" as "mobile phone". Fortunately, plenty of comparative questions are posted online, which provide evidences for what people want to compare, e.g. "*Which to buy, iPod or iPhone*?". We call "*iPod*" and "*iPhone*" in this example as *comparators*. In this paper, we define comparative questions and comparators as:

- **Comparative question**: A question that intends to compare two or more entities and it has to mention these entities explicitly in the question.
- **Comparator**: An entity which is a target of comparison in a comparative question.

According to these definitions, Q1 and Q2 below are not comparative questions while Q3 is. "*iPod Touch*" and "*Zune HD*" are comparators.

Q1: "Which one is better?"
Q2: "Is Lumix GH-1 the best camera?"
Q3: "What's the difference between iPod Touch and Zune HD?"

The goal of this work is mining comparators from comparative questions. The results would be very useful in helping users' exploration of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 650–658,
Uppsala, Sweden, 11-16 July 2010.

alternative choices by suggesting comparable entities based on other users' prior requests.

To mine comparators from comparative questions, we first have to detect whether a question is comparative or not. According to our definition, a comparative question has to be a question with intent to compare at least two entities. Please note that a question containing at least two entities is *not* a comparative question if it does not have comparison intent. However, we observe that a question is very likely to be a comparative question if it contains at least two entities. We leverage this insight and develop a weakly supervised bootstrapping method to identify comparative questions and extract comparators simultaneously.

To our best knowledge, this is the first attempt to specially address the problem on finding good comparators to support users' comparison activity. We are also the first to propose using comparative questions posted online that reflect what users truly care about as the medium from which we mine comparable entities. Our weakly supervised method achieves 82.5% F1-measure in comparative question identification, 83.3% in comparator extraction, and 76.8% in end-to-end comparative question identification and comparator extraction which outperform the most relevant state-of-the-art method by Jindal & Liu (2006b) significantly.

The rest of this paper is organized as follows. The next section discusses previous works. Section 3 presents our weakly-supervised method for comparator mining. Section 4 reports the evaluations of our techniques, and we conclude the paper and discuss future work in Section 5.

2 Related Work

2.1 Overview

In terms of discovering related items for an entity, our work is similar to the research on recommender systems, which recommend items to a user. Recommender systems mainly rely on similarities between items and/or their statistical correlations in user log data (Linden et al., 2003). For example, Amazon recommends products to its customers based on their own purchase histories, similar customers' purchase histories, and similarity between products. However, recommending an item is not equivalent to finding a comparable item. In the case of Amazon, the purpose of recommendation is to entice their customers to add more items to their shopping carts by suggesting similar or related items. While in the case of comparison, we would like to help users explore alternatives, i.e. helping them make a decision among comparable items.

For example, it is reasonable to recommend "*iPod speaker*" or "*iPod batteries*" if a user is interested in "*iPod*", but we would not compare them with "*iPod*". However, items that are comparable with "*iPod*" such as "*iPhone*" or "*PSP*" which were found in comparative questions posted by users are difficult to be predicted simply based on item similarity between them. Although they are all music players, "*iPhone*" is mainly a mobile phone, and "*PSP*" is mainly a portable game device. They are similar but also different therefore beg comparison with each other. It is clear that comparator mining and item recommendation are related but not the same.

Our work on comparator mining is related to the research on entity and relation extraction in information extraction (Cardie, 1997; Califf and Mooney, 1999; Soderland, 1999; Radev et al., 2002; Carreras et al., 2003). Specifically, the most relevant work is by Jindal and Liu (2006a and 2006b) on mining comparative sentences and relations. Their methods applied class sequential rules (CSR) (Chapter 2, Liu 2006) and label sequential rules (LSR) (Chapter 2, Liu 2006) learned from annotated corpora to identify comparative sentences and extract comparative relations respectively in the news and review domains. The same techniques can be applied to comparative question identification and comparator mining from questions. However, their methods typically can achieve high precision but suffer from low recall (Jindal and Liu, 2006b) (J&L). However, ensuring high recall is crucial in our intended application scenario where users can issue arbitrary queries. To address this problem, we develop a weakly-supervised bootstrapping pattern learning method by effectively leveraging unlabeled questions.

Bootstrapping methods have been shown to be very effective in previous information extraction research (Riloff, 1996; Riloff and Jones, 1999; Ravichandran and Hovy, 2002; Mooney and Bunescu, 2005; Kozareva et al., 2008). Our work is similar to them in terms of methodology using bootstrapping technique to extract entities with a specific relation. However, our task is different from theirs in that it requires not only extracting entities (comparator extraction) but also ensuring that the entities are extracted from comparative questions (comparative question identification), which is generally not required in IE task.

2.2 Jindal & Liu 2006

In this subsection, we provide a brief summary of the comparative mining method proposed by Jindal and Liu (2006a and 2006b), which is used as baseline for comparison and represents the state-of-the-art in this area. We first introduce the definition of CSR and LSR rule used in their approach, and then describe their comparative mining method. Readers should refer to J&L's original papers for more details.

CSR and LSR

CSR is a classification rule. It maps a sequence pattern $S(s_1 s_2 \ldots s_n)$ to a class *C*. In our problem, *C* is either *comparative* or *non-comparative*. Given a collection of sequences with class information, every CSR is associated to two parameters: *support* and *confidence*. *Support* is the proportion of sequences in the collection containing *S* as a subsequence. *Confidence* is the proportion of sequences labeled as *C* in the sequences containing the *S*. These parameters are important to evaluate whether a CSR is reliable or not.

LSR is a labeling rule. It maps an input sequence pattern $S(s_1 s_2 \ldots s_i \ldots s_n)$ to a labeled sequence $S'(s_1 s_2 \ldots l_i \ldots s_n)$ by replacing one token (s_i) in the input sequence with a designated label (l_i). This token is referred as the anchor. The anchor in the input sequence could be extracted if its corresponding label in the labeled sequence is what we want (in our case, a comparator). LSRs are also mined from an annotated corpus, therefore each LSR also have two parameters: *support* and *confidence*. They are similarly defined as in CSR.

Supervised Comparative Mining Method

J&L treated comparative sentence identification as a classification problem and comparative relation extraction as an information extraction problem. They first manually created a set of 83 keywords such as *beat*, *exceed*, and *outperform* that are likely indicators of comparative sentences. These keywords were then used as pivots to create part-of-speech (POS) sequence data. A manually annotated corpus with class information, i.e. *comparative* or *non-comparative*, was used to create sequences and CSRs were mined. A Naïve Bayes classifier was trained using the CSRs as features. The classifier was then used to identify comparative sentences.

Given a set of comparative sentences, J&L manually annotated two comparators with labels \$ES1 and \$ES2 and the feature compared with label \$FT for each sentence. J&L's method was only applied to noun and pronoun. To differentiate noun and pronoun that are not comparators or features, they added the fourth label \$NEF, i.e. non-entity-feature. These labels were used as pivots together with special tokens l_i & r_j[1] (token position), #start (beginning of a sentence), and *#end* (end of a sentence) to generate sequence data, sequences with single label only and minimum support greater than 1% are retained, and then LSRs were created. When applying the learned LSRs for extraction, LSRs with higher confidence were applied first.

J&L's method have been proved effective in their experimental setups. However, it has the following weaknesses:

- The performance of J&L's method relies heavily on a set of comparative sentence indicative keywords. These keywords were manually created and they offered no guidelines to select keywords for inclusion. It is also difficult to ensure the completeness of the keyword list.
- Users can express comparative sentences or questions in many different ways. To have high recall, a large annotated training corpus is necessary. This is an expensive process.
- Example CSRs and LSRs given in Jindal & Liu (2006b) are mostly a combination of POS tags and keywords. It is a surprise that their rules achieved high precision but low recall. They attributed most errors to POS tagging errors. However, we suspect that their rules might be too specific and overfit their small training set (about 2,600 sentences). We would like to increase recall, avoid overfitting, and allow rules to include discriminative lexical tokens to retain precision.

In the next section, we introduce our method to address these shortcomings.

3 Weakly Supervised Method for Comparator Mining

Our weakly supervised method is a pattern-based approach similar to J&L's method, but it is different in many aspects: Instead of using separate CSRs and LSRs, our method aims to learn se-

[1] l_i marks a token is at the i^{th} position to the left of the pivot and r_j marks a token is at j^{th} position to the right of the pivot where i and j are between 1 and 4 in J&L (2006b).

quential patterns which can be used to identify

Sequential Patterns
<#start which city is better, $C or $C ? #end>
<, $C or $C ? #end>
<#start $C/NN or $C/NN ? #end>
<which NN is better, $C or $C ?>
<which city is JJR, $C or $C ?>
<which NN is JJR, $C or $C ?>
...

Table 1: Candidate indicative extraction pattern (IEP) examples of the question "*which city is better, NYC or Paris?*"

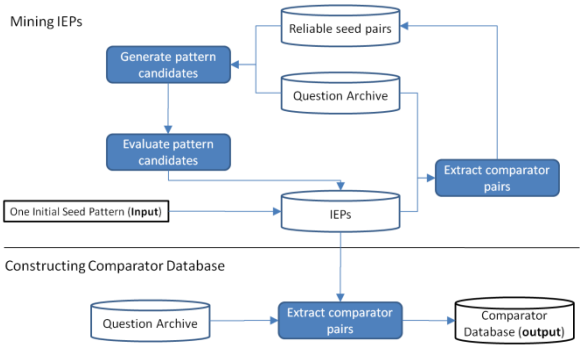


Figure 1: Overview of the bootstrapping alogorithm

comparative question and extract comparators simultaneously.

In our approach, a sequential pattern is defined as a sequence $S(s_1 s_2 \ldots s_i \ldots s_n)$ where s_i can be a word, a POS tag, or a symbol denoting either a comparator ($C), or the beginning (#start) or the end of a question (#end). A sequential pattern is called an *indicative extraction pattern* (IEP) if it can be used to identify comparative questions and extract comparators in them with high reliability. We will formally define the reliability score of a pattern in the next section.

Once a question matches an IEP, it is classified as a comparative question and the token sequences corresponding to the comparator slots in the IEP are extracted as comparators. When a question can match multiple IEPs, the longest IEP is used[2]. Therefore, instead of manually creating a list of indicative keywords, we create a set of IEPs. We will show how to acquire IEPs automatically using a bootstrapping procedure with minimum supervision by taking advantage of a large unlabeled question collection in the following subsections. The evaluations shown in section 4 confirm that our weakly supervised method can achieve high recall while retain high precision.

This pattern definition is inspired by the work of Ravichandran and Hovy (2002). Table 1 shows some examples of such sequential patterns. We also allow POS constraint on comparators as shown in the pattern "<, $C/NN or $C/NN ? #end>". It means that a valid comparator must have a NN POS tag.

3.1 Mining Indicative Extraction Patterns

Our weakly supervised IEP mining approach is based on two key assumptions:

- If a sequential pattern can be used to extract many reliable comparator pairs, it is very likely to be an IEP.
- If a comparator pair can be extracted by an IEP, the pair is *reliable*.

Based on these two assumptions, we design our bootstrapping algorithm as shown in Figure 1. The bootstrapping process starts with a single IEP. From it, we extract a set of initial seed comparator pairs. For each comparator pair, all questions containing the pair are retrieved from a question collection and regarded as comparative questions. From the comparative questions and comparator pairs, all possible sequential patterns are generated and evaluated by measuring their reliability score defined later in the Pattern Evaluation section. Patterns evaluated as reliable ones are IEPs and are added into an IEP repository.

Then, new comparator pairs are extracted from the question collection using the latest IEPs. The new comparators are added to a reliable comparator repository and used as new seeds for pattern learning in the next iteration. All questions from which reliable comparators are extracted are removed from the collection to allow finding new patterns efficiently in later iterations. The process iterates until no more new patterns can be found from the question collection.

There are two key steps in our method: (1) pattern generation and (2) pattern evaluation. In the following subsections, we will explain them in details.

Pattern Generation

To generate sequential patterns, we adapt the surface text pattern mining method introduced in (Ravichandran and Hovy, 2002). For any given comparative question and its comparator pairs, comparators in the question are replaced with symbol $Cs. Two symbols, #start and #end, are attached to the beginning and the end of a sen-

[2] It is because the longest IEP is likely to be the most specific and relevant pattern for the given question.

tence in the question. Then, the following three kinds of sequential patterns are generated from sequences of questions:

- **Lexical patterns:** Lexical patterns indicate sequential patterns consisting of only words and symbols ($C, #start, and #end). They are generated by suffix tree algorithm (Gusfield, 1997) with two constraints: A pattern should contain more than one $C, and its frequency in collection should be more than an empirically determined number β.
- **Generalized patterns:** A lexical pattern can be too specific. Thus, we generalize lexical patterns by replacing one or more words with their POS tags. $2^n - 1$ generalized patterns can be produced from a lexical pattern containing *N* words excluding *$Cs*.
- **Specialized patterns:** In some cases, a pattern can be too general. For example, although a question *"ipod or zune?"* is comparative, the pattern *"*<$C or $C>*"* is too general, and there can be many non-comparative questions matching the pattern, for instance, *"true or false?"*. For this reason, we perform pattern specialization by adding POS tags to all comparator slots. For example, from the lexical pattern "<$C or $C>" and the question "*ipod or zune?*", "<$C/NN or $C/NN?>" will be produced as a specialized pattern.

Note that generalized patterns are generated from lexical patterns and the specialized patterns are generated from the combined set of generalized patterns and lexical patterns. The final set of candidate patterns is a mixture of lexical patterns, generalized patterns and specialized patterns.

Pattern Evaluation

According to our first assumption, a reliability score $R^k(p_i)$ for a candidate pattern p_i at iteration *k* can be defined as follows:

$$R^k(p_i) = \frac{\sum_{\forall cp_j \in CP^{k-1}} N_Q(p_i \rightarrow cp_j)}{N_Q(p_i \rightarrow *)} \quad (1)$$

, where p_i can extract known reliable comparator pairs cp_j. CP^{k-1} indicates the reliable comparator pair repository accumulated until the $(k-1)th$ iteration. $N_Q(x)$ means the number of questions satisfying a condition *x*. The condition $p_i \rightarrow cp_j$ denotes that cp_j can be extracted from a question by applying pattern p_i while the condition $p_i \rightarrow *$ denotes any question containing pattern p_i.

However, Equation (1) can suffer from incomplete knowledge about reliable comparator pairs. For example, very few reliable pairs are generally discovered in early stage of bootstrapping. In this case, the value of Equation (1) might be underestimated which could affect the effectiveness of equation (1) on distinguishing IEPs from non-reliable patterns. We mitigate this problem by a lookahead procedure. Let us denote the set of candidate patterns at the iteration *k* by $\hat{P}^k$. We define the support *S* for comparator pair $\widehat{cp}_i$ which can be extracted by $\hat{P}^k$ and does not exist in the current reliable set:

$$S(\widehat{cp}_i) = N_Q(\hat{P}^k \rightarrow \widehat{cp}_i) \quad (2)$$

where $\hat{P}^k \rightarrow \widehat{cp}_i$ means that one of the patterns in $\hat{P}^k$ can extract $\widehat{cp}_i$ in certain questions. Intuitively, if $\widehat{cp}_i$ can be extracted by many candidate patterns in $\hat{P}^k$, it is likely to be extracted as a reliable one in the next iteration. Based on this intuition, a pair $\widehat{cp}_i$ whose support *S* is more than a threshold α is regarded as a *likely-reliable pair*. Using likely-reliable pairs, lookahead reliability score $\hat{R}(p_i)$ is defined:

$$\hat{R}^k(p_i) = \frac{\sum_{\forall \widehat{cp}_i \in \widehat{CP}^k_{rel}} N_Q(p_i \rightarrow \widehat{cp}_i)}{N_Q(p_i \rightarrow *)} \quad (3)$$

, where $\widehat{CP}^k_{rel}$ indicates a set of likely-reliable pairs based on $\hat{P}^k$.

By interpolating Equation (1) and (3), the final reliability score $R(p_i)^k_{final}$ for a pattern is defined as follows:

$$R(p_i)^k_{final} = \lambda \cdot R^k(p_i) + (1-\lambda) \cdot \hat{R}^k(p_i) \quad (4)$$

Using Equation (4), we evaluate all candidate patterns and select patterns whose score is more than threshold γ as IEPs. All necessary parameter values are empirically determined. We will explain how to determine our parameters in section 4.

4 Experiments

4.1 Experiment Setup

Source Data

All experiments were conducted on about 60M questions mined from Yahoo! Answers' question title field. The reason that we used only a title

field is that they clearly express a main intention of an asker with a form of simple questions in general.

Evaluation Data

Two separate data sets were created for evaluation. First, we collected 5,200 questions by sampling 200 questions from each Yahoo! Answers category[3]. Two annotators were asked to label each question manually as *comparative*, *non-comparative*, or *unknown*. Among them, 139 (2.67%) questions were classified as comparative, 4,934 (94.88%) as non-comparative, and 127 (2.44%) as unknown questions which are difficult to assess. We call this set SET-A.

Because there are only 139 comparative questions in SET-A, we created another set which contains more comparative questions. We manually constructed a keyword set consisting of 53 words such as "*or*" and "*prefer*", which are good indicators of comparative questions. In SET-A, 97.4% of comparative questions contains one or more keywords from the keyword set. We then randomly selected another 100 questions from each Yahoo! Answers category with one extra condition that all questions have to contain at least one keyword. These questions were labeled in the same way as SET-A except that their comparators were also annotated. This second set of questions is referred as SET-B. It contains 853 comparative questions and 1,747 non-comparative questions. For comparative question identification experiments, we used all labeled questions in SET-A and SET-B. For comparator extraction experiments, we used only SET-B. All the remaining unlabeled questions (called as SET-R) were used for training our weakly supervised method.

As a baseline method, we carefully implemented J&L's method. Specifically, CSRs for comparative question identification were learned from the labeled questions, and then a statistical classifier was built by using CSR rules as features. We examined both SVM and Naïve Bayes (NB) models as reported in their experiments. For the comparator extraction, LSRs were learned from SET-B and applied for comparator extraction.

To start the bootstrapping procedure, we applied the IEP *"<#start nn/$c vs/cc nn/$c ?/. #end>"* to all the questions in SET-R and gathered 12,194 comparator pairs as the initial seeds. For our weakly supervised method, there are four parameters, i.e. α, β, γ, and λ, need to be determined empirically. We first mined all possible candidate patterns from the suffix tree using the initial seeds. From these candidate patterns, we applied them to SET-R and got a new set of 59,410 candidate comparator pairs. Among these new candidate comparator pairs, we randomly selected 100 comparator pairs and manually classified them into reliable or non-reliable comparators. Then we found α that maximized precision without hurting recall by investigating frequencies of pairs in the labeled set. By this method, α was set to 3 in our experiments. Similarly, the threshold parameters β and γ for pattern evaluation were set to 10 and 0.8 respectively. For the interpolation parameter λ in Equation (3), we simply set the value to 0.5 by assuming that two reliability scores are equally important.

As evaluation measures for comparative question identification and comparator extraction, we used precision, recall, and F1-measure. All results were obtained from 5-fold cross validation. Note that J&L's method needs a training data but ours use the unlabeled data (SET-R) with weakly supervised method to find parameter setting. This 5-fold evaluation data is not in the unlabeled data. Both methods were tested on the same test split in the 5-fold cross validation. All evaluation scores are averaged across all 5 folds.

For question processing, we used our own statistical POS tagger developed in-house[4].

4.2 Experiment Results

Comparative Question Identification and Comparator Extraction

Table 2 shows our experimental results. In the table, "Identification only" indicates the performances in comparative question identification, "Extraction only" denotes the performances of comparator extraction when only comparative questions are used as input, and "All" indicates the end-to-end performances when question identification results were used in comparator extraction. Note that the results of J&L's method on our collections are very comparable to what is reported in their paper.

In terms of precision, the J&L's method is competitive to our method in comparative ques-

[3] There are 26 top level categories in Yahoo! Answers.

[4] We used NLC-PosTagger which is developed by NLC group of Microsoft Research Asia. It uses the modified Penn Treebank POS set for its output; for example, NNS (plural nouns), NN (nouns), NP (noun phrases), NPS (plural noun phrases), VBZ (verb, present tense, 3rd person singular), JJ (adjective), RB(adverb), and so on.

	Identification only (SET-A+SET-B)			Extraction only (SET-B)		All (SET-B)		
	J&L (CSR)		**Our**	J&L	**Our**	J&L		**Our**
	SVM	NB	**Method**	(LSR)	**Method**	SVM	NB	**Method**
Recall	0.601	0.537	**0.817***	0.621	**0.760***	0.373	0.363	**0.760***
Precision	0.847	**0.851**	0.833	0.861	**0.916***	0.729	0.703	**0.776***
F-score	0.704	0.659	**0.825***	0.722	**0.833***	0.493	0.479	**0.768***

Table 2: Performance comparison between our method and Jindal and Bing's Method (denoted as J&L). The values with * indicate statistically significant improvements over J&L (CSR) SVM or J&L (LSR) according to *t*-test at $p < 0.01$ level.

tion identification. However, the recall is significantly lower than ours. In terms of recall, our method outperforms J&L's method by 35% and 22% in comparative question identification and comparator extraction respectively. In our analysis, the low recall of J&L's method is mainly caused by low coverage of learned CSR patterns over the test set.

In the end-to-end experiments, our weakly supervised method performs significantly better than J&L's method. Our method is about 55% better in F1-measure. This result also highlights another advantage of our method that identifies comparative questions and extracts comparators simultaneously using one single pattern. J&L's method uses two kinds of pattern rules, i.e. CSRs and LSRs. Its performance drops significantly due to error propagations. F1-measure of J&L's method in "All" is about 30% and 32% worse than the scores of "Identification only" and "Extraction" only respectively, our method only shows small amount of performance decrease (approximately 7-8%).

We also analyzed the effect of pattern generalization and specialization. Table 3 shows the results. Despite of the simplicity of our methods, they significantly contribute to performance improvements. This result shows the importance of learning patterns flexibly to capture various comparative question expressions. Among the 6,127 learned IEPs in our database, 5,930 patterns are generalized ones, 171 are specialized ones, and only 26 patterns are non-generalized and specialized ones.

To investigate the robustness of our bootstrapping algorithm for different seed configurations, we compare the performances between two different seed IEPs. The results are shown in Table 4. As shown in the table, the performance of our bootstrapping algorithm is stable regardless of significantly different number of seed pairs generated by the two IEPs. This result implies that our bootstrapping algorithm is not sensitive to the choice of IEP.

Table 5 also shows the robustness of our bootstrapping algorithm. In Table 5, '*All'* indicates the performances that all comparator pairs from a single seed IEP is used for the bootstrapping, and 'Partial' indicate the performances using only 1,000 randomly sampled pairs from '*All'*. As shown in the table, there is no significant performance difference.

In addition, we conducted error analysis for the cases where our method fails to extract correct comparator pairs:

- 23.75% of errors on comparator extraction are due to wrong pattern selection by our simple maximum IEP length strategy.
- The remaining 67.63% of errors come from comparative questions which cannot be covered by the learned IEPs.

	Recall	Precision	F-score
Original Patterns	0.689	0. 449	0.544
+ Specialized	0.731	0.602	0.665
+ Generalized	0.760	0.776	0.768

Table 3: Effect of pattern specialization and Generalization in the end-to-end experiments.

Seed patterns	# of resulted seed pairs	F-score
<#start nn/$c vs/cc nn/$c ?/. #end>	12,194	0.768
<#start which/wdt is/vb better/jjr , nn/$c or/cc nn/$c ?/. #end>	1,478	0.760

Table 4: Performance variation over different initial seed IEPs in the end-to-end experiments

Set (# of seed pairs)	Recall	Precision	F-score
All (12,194)	0.760	0.774	0.768
Partial (1,000)	0.724	0.763	0.743

Table 5: Performance variation over different sizes of seed pairs generated from a single initial seed IEP "*<#start nn/$c vs/cc nn/$c ?/. #end>*".

	Chanel	Gap	iPod	Kobe	Canon
1	Dior	Old Navy	Zune	Lebron	Nikon
2	Louis Vuitton	American Eagle	mp3 player	Jordan	Sony
3	Coach	Banana Republic	PSP	MJ	Kodak
4	Gucci	Guess by Marciano	cell phone	Shaq	Panasonic
5	Prada	ACP Ammunition	iPhone	Wade	Casio
6	Lancome	Old Navy brand	Creative Zen	T-mac	Olympus
7	Versace	Hollister	Zen	Lebron James	Hp
8	LV	Aeropostal	iPod nano	Nash	Lexmark
9	Mac	American Eagle outfitters	iPod touch	KG	Pentax
10	Dooney	Guess	iRiver	Bonds	Xerox

Table 6: Examples of comparators for different entities

Chanel	Gap	iPod	Kobe	Canon
Chanel handbag	Gap coupons	***iPod nano***	Kobe Bryant stats	Canon t2i
Chanel sunglass	Gap outlet	***iPod touch***	Lakers Kobe	Canon printers
Chanel earrings	Gap card	iPod best buy	Kobe espn	Canon printer drivers
Chanel watches	Gap careers	iTunes	Kobe Dallas Mavericks	Canon downloads
Chanel shoes	Gap casting call	Apple	Kobe NBA	Canon copiers
Chanel jewelry	Gap adventures	iPod shuffle	Kobe 2009	Canon scanner
Chanel clothing	***Old navy***	iPod support	Kobe san Antonio	Canon lenses
Dior	***Banana republic***	iPod classic	Kobe Bryant 24	***Nikon***

Table 7: Related queries returned by Google related searches for the same target entities in Table 6. The bold ones indicate overlapped queries to the comparators in Table 6.

Examples of Comparator Extraction

By applying our bootstrapping method to the entire source data (60M questions), 328,364 unique comparator pairs were extracted from 679,909 automatically identified comparative questions.

Table 6 lists top 10 frequently compared entities for a target item, such as *Chanel*, *Gap*, in our question archive. As shown in the table, our comparator mining method successfully discovers realistic comparators. For example, for '*Chanel'*, most results are high-end fashion brands such as '*Dior'* or '*Louis Vuitton'*, while the ranking results for '*Gap'* usually contains similar apparel brands for young people, such as '*Old Navy'* or '*Banana Republic'*. For the basketball player '*Kobe*', most of the top ranked comparators are also famous basketball players. Some interesting comparators are shown for '*Canon*' (the company name). It is famous for different kinds of its products, for example, digital cameras and printers, so it can be compared to different kinds of companies. For example, it is compared to '*HP'*, '*Lexmark'*, or '*Xerox',* the printer manufacturers, and also compared to '*Nikon'*, '*Sony'*, or '*Kodak',* the digital camera manufactures. Besides general entities such as a brand or company name, our method also found an interesting comparable entity for a specific item in the experiments. For example, our method recommends '*Nikon d40i*', '*Canon rebel xti*', '*Canon rebel xt*', '*Nikon d3000*', '*Pentax k100d*', '*Canon eos 1000d*' as comparators for the specific camera product '*Nikon 40d*'.

Table 7 can show the difference between our comparator mining and query/item recommendation. As shown in the table, 'Google related searches' generally suggests a mixed set of two kinds of related queries for a target entity: (1) queries specified with subtopics for an original query (e.g., '*Chanel handbag*' for '*Chanel*') and (2) its comparable entities (e.g., '*Dior*' for '*Chanel*'). It confirms one of our claims that comparator mining and query/item recommendation are related but not the same.

5 Conclusion

In this paper, we present a novel weakly supervised method to identify comparative questions and extract comparator pairs simultaneously. We rely on the key insight that a good comparative question identification pattern should extract good comparators, and a good comparator pair should occur in good comparative questions to bootstrap the extraction and identification process. By leveraging large amount of unlabeled data and the bootstrapping process with slight supervision to determine four parameters, we found 328,364 unique comparator pairs and 6,869 extraction patterns without the need of creating a set of comparative question indicator keywords.

The experimental results show that our method is effective in both comparative question identification and comparator extraction. It sig-

nificantly improves recall in both tasks while maintains high precision. Our examples show that these comparator pairs reflect what users are really interested in comparing.

Our comparator mining results can be used for a commerce search or product recommendation system. For example, automatic suggestion of comparable entities can assist users in their comparison activities before making their purchase decisions. Also, our results can provide useful information to companies which want to identify their competitors.

In the future, we would like to improve extraction pattern application and mine rare extraction patterns. How to identify comparator aliases such as '*LV'* and '*Louis Vuitton*' and how to separate ambiguous entities such "*Paris vs. London*" as location and "*Paris vs. Nicole*" as celebrity are all interesting research topics. We also plan to develop methods to summarize answers pooled by a given comparator pair.

6 Acknowledgement

This work was done when the first author worked as an intern at Microsoft Research Asia.

References

Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI'99 /IAAI'99*.

Claire Cardie. 1997. Empirical methods in information extraction. *AI magazine*, 18:65–79.

Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of WWW '02*, pages 517–526.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of WWW '03*, pages 271–279.

Nitin Jindal and Bing Liu. 2006a. Identifying comparative sentences in text documents. In *Proceedings of SIGIR '06*, pages 244–251.

Nitin Jindal and Bing Liu. 2006b. Mining comparative sentences and relations. In *Proceedings of AAAI '06*.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

Greg Linden, Brent Smith and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, pages 76-80.

Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. *ACM SIGKDD Exploration Newsletter*, 7(1):3–10.

Dragomir Radev, Weiguo Fan, Hong Qi, and Harris Wu and Amardeep Grewal. 2002. Probabilistic question answering on the web. *Journal of the American Society for Information Science and Technology*, pages 408–419.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL '02*, pages 41–47.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI '99 /IAAI '99*, pages 474–479.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1044–1049.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.

Towards robust multi-tool tagging. An OWL/DL-based approach

Christian Chiarcos
University of Potsdam, Germany
chiarcos@uni-potsdam.de

Abstract

This paper describes a series of experiments to test the hypothesis that the parallel application of multiple NLP tools and the integration of their results improves the correctness and robustness of the resulting analysis.

It is shown how annotations created by seven NLP tools are mapped onto tool-independent descriptions that are defined with reference to an ontology of linguistic annotations, and how a majority vote and ontological consistency constraints can be used to integrate multiple alternative analyses of the same token in a consistent way.

For morphosyntactic (parts of speech) and morphological annotations of three German corpora, the resulting merged sets of ontological descriptions are evaluated in comparison to (ontological representation of) existing reference annotations.

1 Motivation and overview

NLP systems for higher-level operations or complex annotations often integrate redundant modules that provide alternative analyses for the same linguistic phenomenon in order to benefit from their respective strengths and to compensate for their respective weaknesses, e.g., in parsing (Crysmann et al., 2002), or in machine translation (Carl et al., 2000). The current trend to parallel and distributed NLP architectures (Aschenbrenner et al., 2006; Gietz et al., 2006; Egner et al., 2007; Luís and de Matos, 2009) opens the possibility of exploring the potential of redundant parallel annotations also for lower levels of linguistic analysis.

This paper evaluates the potential benefits of such an approach with respect to morphosyntax (parts of speech, pos) and morphology in German: In comparison to English, German shows a rich and polysemous morphology, and a considerable number of NLP tools are available, making it a promising candidate for such an experiment.

Previous research indicates that the integration of multiple part of speech taggers leads to more accurate analyses. So far, however, this line of research focused on tools that were trained on the same corpus (Brill and Wu, 1998; Halteren et al., 2001), or that specialize to different subsets of the same tagset (Zavrel and Daelemans, 2000; Tufiş, 2000; Borin, 2000). An even more substantial increase in accuracy and detail can be expected if tools are combined that make use of different annotation schemes.

For this task, ontologies of linguistic annotations are employed to assess the linguistic information conveyed in a particular annotation and to integrate the resulting ontological descriptions in a consistent and tool-independent way. The merged set of ontological descriptions is then evaluated with reference to morphosyntactic and morphological annotations of three corpora of German newspaper articles, the NEGRA corpus (Skut et al., 1998), the TIGER corpus (Brants et al., 2002) and the Potsdam Commentary Corpus (Stede, 2004, PCC).

2 Ontologies and annotations

Various repositories of linguistic annotation terminology have been developed in the last decades, ranging from early texts on annotation standards (Bakker et al., 1993; Leech and Wilson, 1996) over relational data base models (Bickel and Nichols, 2000; Bickel and Nichols, 2002) to more recent formalizations in OWL/RDF (or with OWL/RDF export), e.g., the General Ontology of Linguistic Description (Farrar and Langendoen, 2003, GOLD), the ISO TC37/SC4 Data Category Registry (Ide and Romary, 2004; Kemps-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 659–670,
Uppsala, Sweden, 11-16 July 2010.

Snijders et al., 2009, DCR), the OntoTag ontology (Aguado de Cea et al., 2002), or the Typological Database System ontology (Saulwick et al., 2005, TDS). Despite their common level of representation, however, these efforts have not yet converged into a unified and generally accepted ontology of linguistic annotation terminology, but rather, different resources are maintained by different communities, so that a considerable amount of disagreement between them and their respective definitions can be observed.[1]

Such conceptual mismatches and incompatibilities between existing terminological repositories have been the motivation to develop the OLiA architecture (Chiarcos, 2008) that employs a shallow Reference Model to mediate between (ontological models of) annotation schemes and several existing terminology repositories, incl. GOLD, the DCR, and OntoTag. When an annotation receives a representation in the OLiA Reference Model, it is thus also interpretable with respect to other linguistic ontologies. Therefore, the findings for the OLiA Reference Model in the experiments described below entail similar results for an application of GOLD or the DCR to the same task.

2.1 The OLiA ontologies

The Ontologies of Linguistic Annotations – briefly, OLiA ontologies (Chiarcos, 2008) – represent an architecture of modular OWL/DL ontologies that formalize several intermediate steps of the mapping between concrete annotations, a Reference Model and existing terminology repositories ('External Reference Models' in OLiA terminology) such as the DCR.[2]

The OLiA ontologies were originally developed as part of an infrastructure for the sustainable maintenance of linguistic resources (Schmidt et al., 2006) where they were originally applied to the formal representation and documentation of annotation schemes, and for concept-based annotation queries over to multiple, heterogeneous corpora annotated with different annotation schemes (Rehm et al., 2007; Chiarcos et al., 2008). NLP applications of the OLiA ontologies include a proposal to integrate them with the OntoTag ontologies and to use them for interface specifications between modules in NLP pipeline architectures (Buyko et al., 2008). Further, Hellmann (2010) described the application of the OLiA ontologies within NLP2RDF, an OWL-based blackboard approach to assess the meaning of text from grammatical analyses and subsequent enrichment with ontological knowledge sources.

OLiA distinguishes three different classes of ontologies:

- The OLiA REFERENCE MODEL specifies the common terminology that different annotation schemes can refer to. It is primarily based on a blend of concepts of EAGLES and GOLD, and further extended in accordance with different annotation schemes, with the TDS ontology and with the DCR (Chiarcos, 2010).
- Multiple OLiA ANNOTATION MODELs formalize annotation schemes and tag sets. Annotation Models are based on the original documentation and data samples, so that they provide an authentic representation of the annotation not biased with respect to any particular interpretation.
- For every Annotation Model, a LINKING MODEL defines `subClassOf` ($\sqsubseteq$) relationships between concepts/properties in the respective Annotation Model and the Reference Model. Linking Models are interpretations of Annotation Model concepts and properties in terms of the Reference Model, and thus multiple alternative Linking Models for the same Annotation Model are possible. Other Linking Models specify $\sqsubseteq$ relationships between Reference Model concepts/properties and concepts/properties of an External Reference Model such as GOLD or the DCR.

The OLiA Reference Model (namespace `olia`) specifies concepts that describe linguistic categories (e.g., `olia:Determiner`) and grammatical features (e.g., `olia:Accusative`), as well

[1] As one example, a GOLD Numeral is a Determiner (Numeral $\sqsubseteq$ Quantifier $\sqsubseteq$ Determiner, `http://linguistics-ontology.org/gold/2008/Numeral`), whereas a DCR Numeral is defined on the basis of its semantic function, without any references to syntactic categories (`http://www.isocat.org/datcat/DC-1334`). Thus, *two* in *two of them* is a DCR Numeral but not a GOLD Numeral.

[2] The OLiA Reference Model is accessible via `http://nachhalt.sfb632.uni-potsdam.de/owl/olia.owl`. Several annotation models, e.g., `stts.owl`, `tiger.owl`, `connexor.owl`, `morphisto.owl` can be found in the same directory together with the corresponding linking files `stts-link.rdf`, `tiger-link.rdf`, `connexor-link.rdf` and `morphisto-link.rdf`.

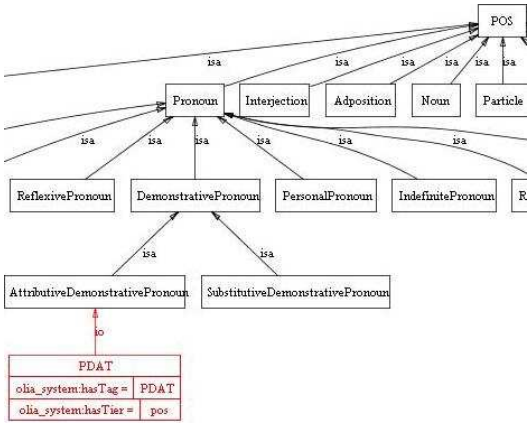


Figure 1: Attributive demonstrative pronouns (`PDAT`) in the STTS Annotation Model

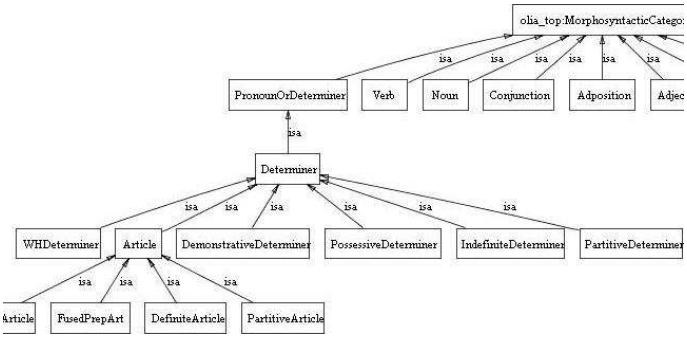


Figure 2: Selected morphosyntactic categories in the OLiA Reference Model

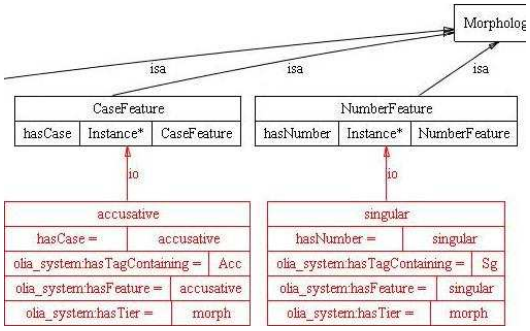


Figure 3: Individuals for accusative and singular in the TIGER Annotation Model

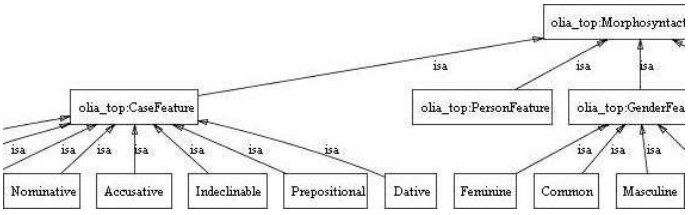


Figure 4: Selected morphological features in the OLiA Reference Model

as properties that define possible relations between those (e.g., `olia:hasCase`). More general concepts that represent organizational information rather than possible annotations (e.g., `MorphosyntacticCategory` and `CaseFeature`) are stored in a separate ontology (namespace `olia_top`).

The Reference Model is a shallow ontology: It does not specify disjointness conditions of concepts and cardinality or domain restrictions of properties. Instead, it assumes that such constraints are inherited by means of $\sqsubseteq$ relationships from an External Reference Model. Different External Reference Models may take different positions on the issue – as languages do[3] –, so that this aspect is left underspecified in the Reference Model.

Figs. 2 and 4 show excerpts of category and feature hierarchies in the Reference Model.

With respect to morphosyntactic annotations (parts of speech, `pos`) and morphological annotations (`morph`), five Annotation Models for German are currently available: STTS (Schiller et al., 1999, `pos`), TIGER (Brants and Hansen, 2002, `morph`), Morphisto (Zielinski and Simon, 2008, `pos, morph`), RFTagger (Schmid and Laws, 2008, `pos, morph`), Connexor (Tapanainen and Järvinen, 1997, `pos, morph`). Further Annotation Models for `pos` and `morph` cover five different annotation schemes for English (Marcus et al., 1994; Sampson, 1995; Mandel, 2006; Kim et al., 2003, Connexor), two annotation schemes for Russian (Meyer, 2003; Sharoff et al., 2008), an annotation scheme designed for typological research and currently applied to approx. 30 different languages (Dipper et al., 2007), an annotation scheme for Old High German (Petrova et al., 2009), and an annotation scheme for Tibetan (Wagner and Zeisler, 2004).

[3]Based on primary experience with Western European languages, for example, one might assume that a `hasGender` property applies to nouns, adjectives, pronouns and determiners only. Yet, this is language-specific restriction: Russian finite verbs, for example, show gender congruency in past tense.

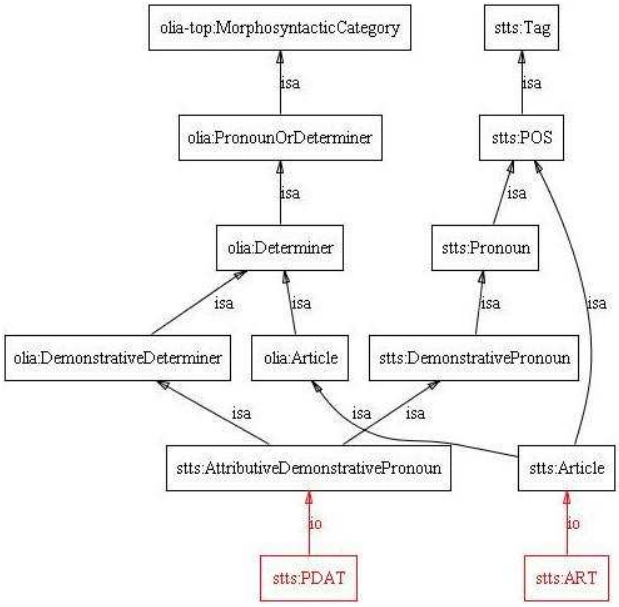


Figure 5: The STTS tags `PDAT` and `ART`, their representation in the Annotation Model and linking with the Reference Model.

Annotation Models differ from the Reference Model mostly in that they include not only concepts and properties, but also individuals: Annotation Model concepts reflect an abstract conceptual categorization, whereas individuals represent concrete values used to annotate the corresponding phenomenon. An individual is applicable to all annotations that match the string value specified by this individual's `hasTag`, `hasTagContaining`, `hasTagStartingWith`, or `hasTagEndingWith` properties. Fig. 1 illustrates the structure of the STTS Annotation Model (namespace `stts`) for the individual `stts:PDAT` that represents the tag used for attributive demonstrative pronouns (demonstrative determiners). Fig. 3 illustrates the individuals `tiger:accusative` and `tiger:singular` from the hierarchy of morphological features in the TIGER Annotation Model (namespace `tiger`).

Fig. 5 illustrates the linking between the STTS Annotation Model and the OLiA Reference Model for the individuals `stts:PDAT` and `stts:ART`.

2.2 Integrating different morphosyntactic and morphological analyses

With the OLiA ontologies as described above, annotations from different annotation schemes can now be interpreted in terms of the OLiA Reference Model (or External Reference Models like GOLD or the DCR).

As an example, consider the attributive demonstrative pronoun *diese* in (1).

(1)

Diese	nicht	neue	Erkenntnis	konnte
this	not	new	insight	could
der	Markt	der	Möglichkeiten	am
the	market	of.the	possibilities	on.the
Sonnabend	in	Treuenbrietzen	bestens	
Saturday	in	Treuenbrietzen	in.the.best.way	
unterstreichen	.			
underline				

'The 'Market of Possibilities', held this Saturday in Treuenbrietzen, provided best evidence for this well-known (lit. 'not new') insight.' (PCC, #4794)

The phrase *diese nicht neue Erkenntnis* poses two challenges. First, it has to be recognized that the demonstrative pronoun is attributive, although it is separated from adjective and noun by *nicht* 'not'. Second, the phrase is in accusative case, although the morphology is ambiguous between accusative and nominative, and nominative case would be expected for a sentence-initial NP.

The Connexor analysis (Tapanainen and Järvinen, 1997) actually fails in both aspects (2).

(2) `PRON Dem FEM SG NOM` (Connexor)

The ontological analysis of this annotation begins by identifying the set of individuals from the Connexor Annotation Model that match it according to their `hasTag` (etc.) properties. The RDF triplet `connexor:NOM connexor:hasTagContaining 'NOM'`[4] indicates that the tag is an application of the individual `connexor:NOM`, an instance of `connexor:Case`. Further, the annotation matches `connexor:PRON` (an instance of `connexor:Pronoun`), etc. The result is a set of individuals that express different aspects of the meaning of the annotation.

For these individuals, the Annotation Model specifies superclasses (`rdf:type`) and other properties, i.e., `connexor:NOM connexor:hasCase connexor:NOM`, etc. The linguistic unit represented by the actual token can now be characterized by these properties: Every property applicable to a member in the individual set is assumed to be applicable to the linguistic unit as well. In order to save space, we use a notation closer to predicate logic (with the token as implicit subject). In terms of the Annotation Model, the token *diese* is thus described by the following descriptions:

[4]RDF triplets are quoted in simplified form, with XML namespaces replacing the actual URIs.

(3) `rdf:type(connexor:Pronoun)`
`connexor:hasCase(connexor:NOM) ...`

The Linking Model `connexor-link.rdf` provides us with the information that (i) `connexor:Pronoun` is a subclass of the Reference Model concept `olia:Pronoun`, (ii) `connexor:NOM` is an instance of the Reference Model concept `olia:Nominative`, and (iii) `olia:hasCase` is a subproperty of `olia:hasCase`.

Accordingly, the predicates that describe the token *diese* can be reformulated in terms of the Reference Model. `rdf:type(connexor:Pronoun)` entails `rdf:type(olia:Pronoun)`, etc. Similarly, we know that for some i:`olia:Nominative` it is true that `olia:hasCase(`i`)`, abbreviated here as `olia:hasCase(some olia:Nominative)`.

In this way, the grammatical information conveyed in the original Connexor annotation can be represented in an annotation-independent and tagset-neutral way as shown for the Connexor analysis in (4).

(4) `rdf:type(olia:PronounOrDeterminer)`
`rdf:type(olia:Pronoun)`
`olia:hasNumber(some olia:Singular)`
`olia:hasGender(some olia:Feminine)`
`rdf:type(olia:DemonstrativePronoun)`
`olia:hasCase(some olia:Nominative)`

Analogously, the corresponding RFTagger analysis (Schmid and Laws, 2008) given in (5) can be transformed into a description in terms of the OLiA Reference Model such as in (6).

(5) `PRO.Dem.Attr.-3.Acc.Sg.Fem` (RFTagger)

(6) `rdf:type(olia:PronounOrDeterminer)`
`olia:hasNumber(some olia:Singular)`
`olia:hasGender(some olia:Feminine)`
`olia:hasCase(some olia:Accusative)`
`rdf:type(olia:DemonstrativeDeterminer)`
`rdf:type(olia:Determiner)`

For every description obtained from these (and further) analyses, an integrated and consistent generalization can be established as described in the following section.

3 Processing linguistic annotations

3.1 Evaluation setup

Fig. 6 sketches the architecture of the evaluation environment set up for this study.[5] The input to the system is a set of documents with

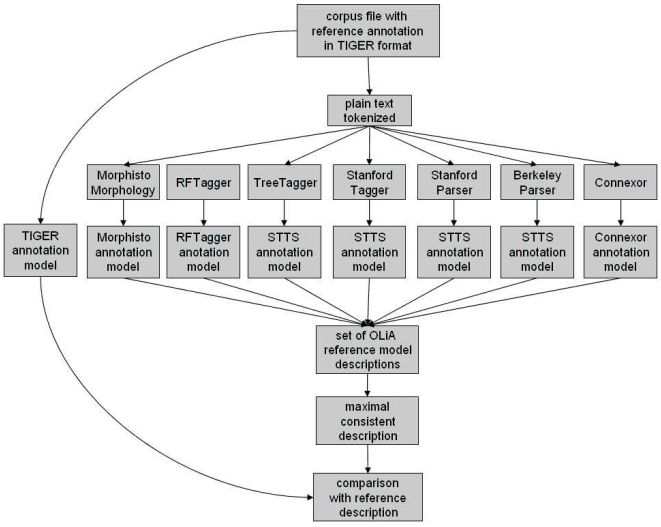


Figure 6: Evaluation setup

TIGER/NEGRA-style morphosyntactic or morphological annotation (Skut et al., 1998; Brants and Hansen, 2002) whose annotations are used as gold standard.

From the annotated document, the plain tokenized text is extracted and analyzed by one or more of the following NLP tools:

(i) Morphisto, a morphological analyzer without contextual disambiguation (Zielinski and Simon, 2008),

(ii) two part of speech taggers: the TreeTagger (Schmid, 1994) and the Stanford Tagger (Toutanova et al., 2003),

(iii) the RFTagger that performs part of speech and morphological analysis (Schmid and Laws, 2008),

(iv) two PCFG parsers: the StanfordParser (Klein and Manning, 2003) and the BerkeleyParser (Petrov and Klein, 2007), and

(v) the Connexor dependency parser (Tapanainen and Järvinen, 1997).

These tools annotate parts of speech, and those in (i), (iii) and (v) also provide morphological features. All components ran in parallel threads on the same machine, with the exception of Morphisto that was addressed as a web service. The set of matching Annotation Model individuals for every annotation and the respective set of Reference Model descriptions are determined by means of

[5] The code used for the evaluation setup is available under `http://multiparse.sourceforge.net`.

OLiA description	$\sum$	Morphisto	Connexor	RF Tagger	Tree Tagger	Stanford Tagger	Stanford Parser	Berkeley Parser
word class `type(...)`								
`PronounOrDeterminer`	7	1(4/4)*	1	1	1	1	1	1
`Determiner`	5.5	0.5**	0	1	1	1	1	1
`DemonstrativeDeterminer`	5.5	0.5**	0	1	1	1	1	1
`Pronoun`	1.5	0.5**	1	0	0	0	0	0
`DemonstrativePronoun`	1.5	0.5**	1	0	0	0	0	0
morphology `hasXY(...)`					n/a	n/a	n/a	n/a
`hasNumber(some Singular)`	2.5	0.5 (2/4)	1	1				
`hasGender(some Feminine)`	2.5	0.5 (2/4)	1	1				
`hasCase(some Accusative)`	1.5	0.5 (2/4)	0	1				
`hasCase(some Nominative)`	1.5	0.5 (2/4)	1	0				
`hasNumber(some Plural)`	0.5	0.5 (2/4)	0	0				

* Morphisto produces four alternative candidate analyses for this example, so every alternative analysis receives the confidence score 0.25

** Morphisto does not distinguish attributive and substitutive pronouns, it predicts `type(Determiner ⊔ Pronoun)`

Table 1: Confidence scores for *diese* in ex. (1)

the Pellet reasoner (Sirin et al., 2007) as described above.

A disambiguation routine (see below) then determines the maximal consistent set of ontological descriptions. Finally, the outcome of this process is compared to the set of descriptions corresponding to the original annotation in the corpus.

3.2 Disambiguation

Returning to examples (4) and (6) above, we see that the resulting set of descriptions conveys properties that are obviously contradicting, e.g., `hasCase(some Nominative)` besides `hasCase(some Accusative)`.

Our approach to disambiguation combines ontological consistency criteria with a confidence ranking. As we simulate an uninformed approach, the confidence ranking follows a majority vote.

For *diese* in (1), the consultation of all seven tools results a confidence ranking as shown in Tab. 1: If a tool supports a description with its analysis, the confidence score is increased by 1 (or by $1/n$ if the tool proposes n alternative annotations). A maximal consistent set of descriptions is then established as follows:

(i) Given a confidence-ranked list of available descriptions $S = (s_1, ..., s_n)$ and a result set $T = \emptyset$.

(ii) Let s_1 be the first element of $S = (s_1, ..., s_n)$.

(iii) If s_1 is consistent with every description $t \in T$, then add s_1 to T: $T := T \cup \{s_1\}$

(iv) Remove s_1 from S and iterate in (ii) until S is empty.

The consistency of ontological descriptions is defined here as follows:[6]

- Two concepts A and B are consistent iff

  $$A \equiv B \text{ or } A \sqsubseteq B \text{ or } B \sqsubseteq A$$

  Otherwise, A and B are disjoint.

- Two descriptions $pred_1(A)$ and $pred_2(B)$ are consistent iff

  A and B are consistent or
  $pred_1$ is neither a subproperty
  nor a superproperty of $pred_2$

This heuristic formalizes an implicit disjointness assumption for all concepts in the ontology (all concepts are disjoint unless one is a subconcept of the other). Further, it imposes an implicit cardinality constraint on properties (e.g., `hasCase(some Accusative)` and `hasCase(some Nominative)` are inconsistent because `Accusative` and `Nominative` are sibling concepts and thus disjoint).

For the example *diese*, the descriptions `type(Pronoun)` and `type(DemonstrativePronoun)` are inconsistent with `type(Determiner)`, and `hasNumber(some Plural)` is inconsistent with `hasNumber(some Singular)` (Figs. 2 and 4); these descriptions are thus ruled out. The `hasCase` descriptions have identical confidence scores, so that the first `hasCase` description that the algorithm encounters is chosen for the set of resulting descriptions, the other one is ruled out because of their inconsistency.

[6]The OLiA Reference Model does not specify disjointness constraints, and neither do GOLD or the DCR as External Reference Models. The axioms of the OntoTag ontologies, however, are specific to Spanish and cannot be directly applied to German.

	PCC	TIGER	NEGRA
best-performing tool (StanfordTagger)			
	.960	.956	.990*
average (and std. deviation) for tool combinations			
1 tool	.868 (.109)	.864 (.122)	.870 (.113)
2 tools	.928 (.018)	.931 (.021)	.943 (.028)
3 tools	.947 (.014)	.948 (.013)	.956 (.018)
4 tools	.956 (.006)	.955 (.009)	.963 (.013)
5 tools	.959 (.006)	.960 (.007)	.964 (.009)
6 tools	.963 (.003)	.963 (.007)	.965 (.007)
all tools	.967	.960	.965

* The Stanford Tagger was trained on the NEGRA corpus.

Table 2: Recall for `rdf:type` descriptions for word classes

	TIGER	NEGRA
1 tool	.678 (.106)	.660 (.091)
Morphisto	.573	.568
Connexor	.674	.662
RFTagger	.786	.751
2 tools	.761 (.019)	.740 (.012)
C+M	.738	.730
M+R	.769	.737
C+R	.773	.753
all tools	.791	.770

Table 3: Recall for morphological `hasXY()` descriptions

The resulting, maximal consistent set of descriptions is then compared with the ontological descriptions that correspond to the original annotation in the corpus.

4 Evaluation

Six experiments were conducted with the goal to evaluate the prediction of word classes and morphological features on parts of three corpora of German newspaper articles: NEGRA (Skut et al., 1998), TIGER (Brants et al., 2002), and the Potsdam Commentary Corpus (Stede, 2004, PCC). From every corpus 10,000 tokens were considered for the analysis.

TIGER and NEGRA are well-known resources that also influenced the design of several of the tools considered. For this reason, the PCC was consulted, a small collection of newspaper commentaries, 30,000 tokens in total, annotated with TIGER-style parts of speech and syntax (by members of the TIGER project). None of the tools considered here were trained on this data, so that it provides independent test data.

The ontological descriptions were evaluated for recall:[7]

$$(7)\quad recall(T) = \frac{\sum_{i=1}^{n} |D_{predicted}(t_i) \cap D_{target}(t_i)|}{\sum_{i=1}^{n} |D_{target}(t_i)|}$$

In (7), T is a text (a list of tokens) with $T = (t_1, ..., t_n)$, $D_{predicted}(t)$ are descriptions retrieved from the NLP analyses of the token t, and $D_{target}(t)$ is the set of descriptions that correspond to the original annotation of t in the corpus.

[7] Precision and accuracy may not be appropriate measurements in this case: Annotation schemes differ in their expressiveness, so that a description predicted by an NLP tool but not found in the reference annotation may nevertheless be correct. The RFTagger, for example, assigns demonstrative pronouns the feature '3rd person', that is not found in TIGER/NEGRA-style annotation because of its redundancy.

4.1 Word classes

Table 2 shows that the recall of `rdf:type` descriptions (for word classes) increases continuously with the number of NLP tools applied. The combination of all seven tools actually shows a better recall than the best-performing single NLP tool. (The NEGRA corpus is an apparent exception only; the exceptionally high recall of the Stanford Tagger reflects the fact that it was trained on NEGRA.)

A particularly high increase in recall occurs when tools are combined that compensate for their respective deficits. Morphisto, for example, generates alternative morphological analyses, so that the disambiguation algorithm performs a random choice between these. Morphisto has thus the worst recall among all tools considered (PCC .69, TIGER .65, NEGRA .70 for word classes). As compared to this, Connexor performs a contextual disambiguation; its recall is, however, limited by its coarse-grained word classes (PCC .73, TIGER .72, NEGRA .73). The combination of both tools yields a more detailed and context-sensitive analysis and thus results in a boost in recall by more than 13% (PCC .87, TIGER .86, NEGRA .86).

4.2 Morphological features

For morphological features, Tab. 3 shows the same tendencies that were also observed for word classes: The more tools are combined, the greater the recall of the generated descriptions, and the recall of combined tools often outperforms the recall of individual tools.

The three tools that provide morphological annotations (Morphisto, Connexor, RFTagger) were evaluated against 10,000 tokens from TIGER and NEGRA respectively. The best-performing tool was the RFTagger, which possibly reflects the fact

that it was trained on TIGER-style annotations, whereas Morphisto and Connexor were developed on the basis of independent resources and thus differ from the reference annotation in their respective degree of granularity.

5 Summary and Discussion

With the ontology-based approach described in this paper, the performance of annotation tools can be evaluated on a conceptual basis rather than by means of a string comparison with target annotations. A formal model of linguistic concepts is extensible, finer-grained and, thus, potentially more adequate for the integration of linguistic annotations than string-based representations, especially for heterogeneous annotations, if the tagsets involved are structured according to different design principles (e.g., due to different terminological traditions, different communities involved, etc.).

It has been shown that by **abstracting from tool-specific representations** of linguistic annotations, annotations from different tagsets can be represented with reference to the OLiA ontologies (and/or with other OWL/RDF-based terminology repositories linked as External Reference Models). In particular, it is possible to compare an existing reference annotation with annotations produced by NLP tools that use independently developed and differently structured annotation schemes (such as Connexor vs. RFTagger vs. Morphisto).

Further, an algorithm for the **integration of different annotations** has been proposed that makes use of a majority-based confidence ranking and ontological consistency conditions. As consistency conditions are not formally defined in the OLiA Reference Model (which is expected to inherit such constraints from External Reference Models), a heuristic, structure-based definition of consistency was applied.

This heuristic consistency definition is overly rigid and rules out a number of consistent alternative analyses, as it is the case for overlapping categories.[8] Despite this rigidity, we witness an **increase of recall** when multiple alternative analyses are integrated. This increase of recall may result from a compensation of tool-specific deficits, e.g., with respect to annotation granularity. Also, the improved recall can be explained by a compensation of overfitting, or deficits that are inherent to a particular approach (e.g., differences in the coverage of the linguistic context).

It can thus be stated that the integration of multiple alternative analyses has the potential to produce linguistic analyses that are both more robust and more detailed than those of the original tools.

The primary field of application of this approach is most likely to be seen in a context where applications are designed that make direct use of OWL/RDF representations as described, for example, by Hellmann (2010). It is, however, also possible to use ontological representations to bootstrap novel and more detailed annotation schemes, cf. Zavrel and Daelemans (2000). Further, the conversion from string-based representations to ontological descriptions is reversible, so that results of ontology-based disambiguation and validation can also be reintegrated with the original annotation scheme. The idea of such a reversion algorithm was sketched by Buyko et al. (2008) where the OLiA ontologies were suggested as a means to translate between different annotation schemes.[9]

6 Extensions and Related Research

Natural extensions of the approach described in this paper include:

(i) Experiments with formally defined consistency conditions (e.g., with respect to restrictions on the domain of properties).

(ii) Context-sensitive disambiguation of morphological features (e.g., by combination with a chunker and adjustment of confidence scores for morphological features over all tokens in the current chunk, cf. Kermes and Evert, 2002).

(iii) Replacement of majority vote by more elaborate strategies to merge grammatical analyses.

[8] Preposition-determiner compounds like German *am* 'on the', for example, are both prepositions and determiners.

[9] The mapping from ontological descriptions to tags of a particular scheme is possible, but neither trivial nor necessarily lossless: Information of ontological descriptions that cannot be expressed in the annotation scheme under consideration (e.g., the distinction between attributive and substitutive pronouns in the Morphisto scheme) will be missing in the resulting string representation. For complex annotations, where ontological descriptions correspond to different substrings, an additional 'tag grammar' may be necessary to determine the appropriate ordering of substrings according to the annotation scheme (e.g., in the Connexor analysis).

(iv) Application of the algorithm for the ontological processing of node labels and edge labels in syntax annotations.

(v) Integration with other ontological knowledge sources in order to improve the recall of morphosyntactic and morphological analyses (e.g., for disambiguating grammatical case).

Extensions (iii) and (iv) are currently pursued in an ongoing research effort described by Chiarcos et al. (2010). Like morphosyntactic and morphological features, node and edge labels of syntactic trees are ontologically represented in several Annotation Models, the OLiA Reference Model, and External Reference Models, the merging algorithm as described above can thus be applied for syntax, as well. Syntactic annotations, however, involve the additional challenge to align different structures before node and edge labels can be addressed, an issue not further discussed here for reasons of space limitations.

Alternative strategies to merge grammatical analyses may include alternative voting strategies as discussed in literature on classifier combination, e.g., weighted majority vote, pairwise voting (Halteren et al., 1998), credibility profiles (Tufiş, 2000), or hand-crafted rules (Borin, 2000). A novel feature of our approach as compared to existing applications of these methods is that confidence scores are not attached to plain strings, but to ontological descriptions: Tufiş, for example, assigned confidence scores not to tools (as in a weighted majority vote), but rather, assessed the 'credibility' of a tool *with respect to the predicted tag*. If this approach is applied to ontological descriptions in place of tags, it allows us to consider the credibility of pieces of information regardless of the actual string representation of tags. For example, the credibility of `hasCase` descriptions can be assessed independently from the credibility of `hasGender` descriptions even if the original annotation merged both aspects in one single tag (as the RFTagger does, for example, cf. ex. 5).

Extension (v) has been addressed in previous research, although mostly with the opposite perspective: Already Cimiano and Reyle (2003) noted that the integration of grammatical and semantic analyses may be used to resolve ambiguity and underspecifications, and this insight has also motivated the ontological representation of linguistic resources such as WordNet (Gangemi et al., 2003), FrameNet (Scheffczyk et al., 2006), the linking of corpora with such ontologies (Hovy et al., 2006), the modelling of entire corpora in OWL/DL (Burchardt et al., 2008), and the extension of existing ontologies with ontological representations of selected linguistic features (Buitelaar et al., 2006; Davis et al., 2008).

Aguado de Cea et al. (2004) sketched an architecture for the closer ontology-based integration of grammatical and semantic information using OntoTag and several NLP tools for Spanish. Aguado de Cea et al. (2008) evaluate the benefits of this approach for the Spanish particle *se*, and conclude for this example that the combination of multiple tools yields more detailed and more accurate linguistic analyses of particularly problematic, polysemous function words. A similar increase in accuracy has also been repeatedly reported for ensemble combination approaches, that are, however, limited to tools that produce annotations according to the *same* tagset (Brill and Wu, 1998; Halteren et al., 2001).

These observations provide further support for our conclusion that the ontology-based integration of morphosyntactic analyses enhances both the robustness and the level of detail of morphosyntactic and morphological analyses. Our approach extends the philosophy of ensemble combination approaches to NLP tools that do not only employ different strategies and philosophies, but also different annotation schemes.

Acknowledgements

From 2005 to 2008, the research on linguistic ontologies described in this paper was funded by the German Research Foundation (DFG) in the context of the Collaborative Research Center (SFB) 441 "Linguistic Data Structures", Project C2 "Sustainability of Linguistic Resources" (University of Tübingen), and since 2007 in the context of the SFB 632 "Information Structure", Project D1 "Linguistic Database" (University of Potsdam). The author would also like to thank Julia Ritz, Angela Lahee, Olga Chiarcos and three anonymous reviewers for helpful hints and comments.

References

G. Aguado de Cea, Á. I. de Mon-Rego, A. Pareja-Lora, and R. Plaza-Arteche. 2002. OntoTag: A semantic web page linguistic annotation model. In *Proceedings of the ECAI 2002 Workshop on Semantic Authoring, Annotation and Knowledge Markup*, Lyon, France, July.

G. Aguado de Cea, A. Gomez-Perez, I. Alvarez de Mon, and A. Pareja-Lora. 2004. OntoTag's linguistic ontologies: Improving semantic web annotations for a better language understanding in machines. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, Las Vegas, Nevada, USA, April.

G. Aguado de Cea, J. Puch, and J.Á. Ramos. 2008. Tagging Spanish texts: The problem of "se". In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco, May.

A. Aschenbrenner, P. Gietz, M.W. Küster, C. Ludwig, and H. Neuroth. 2006. TextGrid. A modular platform for collaborative textual editing. In *Proceedings of the International Workshop on Digital Library Goes e-Science (DLSci06)*, pages 27–36, Alicante, Spain, September.

D. Bakker, O. Dahl, M. Haspelmath, M. Koptjevskaja-Tamm, C. Lehmann, and A. Siewierska. 1993. EUROTYP guidelines. Technical report, European Science Foundation Programme in Language Typology.

B. Bickel and J. Nichols. 2000. The goals and principles of AUTOTYP. `http://www.uni-leipzig.de/~autotyp/theory.html`. version of 01/12/2007.

B. Bickel and J. Nichols. 2002. Autotypologizing databases and their use in fieldwork. In *Proceedings of the LREC 2002 Workshop on Resources and Tools in Field Linguistics*, Las Palmas, Spain, May.

L. Borin. 2000. Something borrowed, something blue: Rule-based combination of POS taggers. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece, May, 31st – June, 2nd.

S. Brants and S. Hansen. 2002. Developments in the TIGER annotation scheme and their realization in the corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1643–1649, Las Palmas, Spain, May.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41, Sozopol, Bulgaria, September.

E. Brill and J. Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, pages 191–195, Montréal, Canada, August.

P. Buitelaar, T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel, and P. Cimiano. 2006. LingInfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May.

A. Burchardt, S. Padó, D. Spohr, A. Frank, and U. Heid. 2008. Formalising Multi-layer Corpora in OWL/DL – Lexicon Modelling, Querying and Consistency Control. In *Proceedings of the 3rd International Joint Conference on NLP (IJCNLP 2008)*, Hyderabad, India, January.

E. Buyko, C. Chiarcos, and A. Pareja-Lora. 2008. Ontology-based interface specifications for a NLP pipeline architecture. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco, May.

M. Carl, C. Pease, L.L. Iomdin, and O. Streiter. 2000. Towards a dynamic linkage of example-based and rule-based machine translation. *Machine Translation*, 15(3):223–257.

C. Chiarcos, S. Dipper, M. Götze, U. Leser, A. Lüdeling, J. Ritz, and M. Stede. 2008. A Flexible Framework for Integrating Annotations from Different Tools and Tag Sets. *Traitement Automatique des Langues*, 49(2).

C. Chiarcos, K. Eckart, and J. Ritz. 2010. Creating and exploiting a resource of parallel parses. In *4th Linguistic Annotation Workshop (LAW 2010), held in conjunction with ACL-2010*, Uppsala, Sweden, July.

C. Chiarcos. 2008. An ontology of linguistic annotations. *LDV Forum*, 23(1):1–16. Foundations of Ontologies in Text Technology, Part II: Applications.

C. Chiarcos. 2010. Grounding an ontology of linguistic annotations in the Data Category Registry. In *Workshop on Language Resource and Language Technology Standards (LR&LTS 2010), held in conjunction with LREC 2010*, Valetta, Malta, May.

P. Cimiano and U. Reyle. 2003. Ontology-based semantic construction, underspecification and disambiguation. In *Proceedings of the Lorraine/Saarland Workshop on Prospects and Recent Advances in the Syntax-Semantics Interface*, pages 33–38, Nancy, France, October.

B. Crysmann, A. Frank, B. Kiefer, S. Müller, G. Neumann, J. Piskorski, U. Schäfer, M. Siegel, H. Uszkoreit, F. Xu, M. Becker, and H. Krieger. 2002. An

integrated architecture for shallow and deep processing. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 441–448, Philadelphia, Pennsylvania, USA, July.

B. Davis, S. Handschuh, A. Troussov, J. Judge, and M. Sogrin. 2008. Linguistically light lexical extensions for ontologies. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco, May.

S. Dipper, M. Götze, and S. Skopeteas, editors. 2007. *Information Structure in Cross-Linguistic Corpora: Annotation Guidelines for Phonology, Morphology, Syntax, Semantics, and Information Structure*. Interdisciplinary Studies on Information Structure (ISIS), Working Papers of the SFB 632; 7. Universitätsverlag Potsdam, Potsdam, Germany.

M.T. Egner, M. Lorch, and E. Biddle. 2007. UIMA Grid: Distributed large-scale text analysis. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID'07)*, pages 317–326, Rio de Janeiro, Brazil, May.

S. Farrar and D.T. Langendoen. 2003. Markup and the GOLD ontology. In *EMELD Workshop on Digitizing and Annotating Text and Field Recordings*. Michigan State University, July.

A. Gangemi, R. Navigli, and P. Velardi. 2003. The OntoWordNet project: Extension and axiomatization of conceptual relations in WordNet. In R. Meersman and Z. Tari, editors, *Proceedings of On the Move to Meaningful Internet Systems (OTM2003)*, pages 820–838, Catania, Italy, November.

P. Gietz, A. Aschenbrenner, S. Budenbender, F. Jannidis, M.W. Küster, C. Ludwig, W. Pempe, T. Vitt, W. Wegstein, and A. Zielinski. 2006. TextGrid and eHumanities. In *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (E-SCIENCE '06)*, pages 133–141, Amsterdam, The Netherlands, December.

H. van Halteren, J. Zavrel, and W. Daelmans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, Montréal, Canada, August.

H. van Halteren, J. Zavrel, and W. Daelmans. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229.

S. Hellmann. 2010. The semantic gap of formalized meaning. In *The 7th Extended Semantic Web Conference (ESWC 2010)*, Heraklion, Greece, May 30th – June 3rd.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: the 90% solution. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2006)*, pages 57–60, New York, June.

N. Ide and L. Romary. 2004. A registry of standard data categories for linguistic annotation. In *Proceedings of the Fourth Language Resources and Evaluation Conference (LREC 2004)*, pages 135–39, Lisboa, Portugal, May.

M. Kemps-Snijders, M. Windhouwer, P. Wittenburg, and S.E. Wright. 2009. ISOcat: remodelling metadata for language resources. *International Journal of Metadata, Semantics and Ontologies*, 4(4):261–276.

H. Kermes and S. Evert. 2002. YAC – A recursive chunker for unrestricted German text. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1805–1812, Las Palmas, Spain, May.

J.D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus – A semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):180–182.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July.

G. Leech and A. Wilson. 1996. EAGLES recommendations for the morphosyntactic annotation of corpora. Version of March 1996.

T. Luís and D.M. de Matos. 2009. High-performance high-volume layered corpora annotation. In *Proceedings of the Third Linguistic Annotation Workshop (LAW-III) held in conjunction with ACL-IJCNLP 2009*, pages 99–107, Singapore, August.

M. Mandel. 2006. Integrated annotation of biomedical text: Creating the PennBioIE corpus. In *Text Mining Ontologies and Natural Language Processing in Biomedicine*, Manchester, UK, March.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

R. Meyer. 2003. Halbautomatische morphosyntaktische Annotation russischer Texte. In R. Hammel and L. Geist, editors, *Linguistische Beiträge zur Slavistik aus Deutschland und Österreich. X. JungslavistInnen-Treffen, Berlin 2001*, pages 92–105. Sagner, München.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2007)*, pages 404–411, Rochester, NY, April.

S. Petrova, C. Chiarcos, J. Ritz, M. Solf, and A. Zeldes. 2009. Building and using a richly annotated interlinear diachronic corpus: The case of Old High German Tatian. *Traitement automatique des langues et langues anciennes*, 50(2):47–71.

G. Rehm, R. Eckart, and C. Chiarcos. 2007. An OWL- and XQuery-based mechanism for the retrieval of linguistic patterns from XML-corpora. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, Borovets, Bulgaria, September.

G. Sampson. 1995. *English for the computer: The SUSANNE corpus and analytic scheme*. Oxford University Press.

A. Saulwick, M. Windhouwer, A. Dimitriadis, and R. Goedemans. 2005. Distributed tasking in ontology mediated integration of typological databases for linguistic research. In *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, Porto, Portugal, June.

J. Scheffczyk, A. Pease, and M. Ellsworth. 2006. Linking FrameNet to the suggested upper merged ontology. In *Proceedings of the Fourth International Conference on Formal Ontology in Information Systems (FOIS 2006)*, pages 289–300, Baltimore, Maryland, USA, November.

A. Schiller, S. Teufel, C. Thielen, and C. Stöckert. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, University of Stuttgart, University of Tübingen.

H. Schmid and F. Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained pos tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester, UK, August.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, September.

T. Schmidt, C. Chiarcos, T. Lehmberg, G. Rehm, A. Witt, and E. Hinrichs. 2006. Avoiding data graveyards: From heterogeneous data collected in multiple research projects to sustainable linguistic resources. In *Proceedings of the E-MELD workshop on Digital Language Documentation: Tools and Standards: The State of the Art*, East Lansing, Michigan, US, June.

S. Sharoff, M. Kopotev, T. Erjavec, A. Feldman, and D. Divjak. 2008. Designing and evaluating Russian tagsets. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco, May.

E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. 2007. Pellet: A practical OWL/DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

W. Skut, T. Brants, B. Krenn, and H. Uszkoreit. 1998. A linguistically interpreted corpus of German newspaper text. In *In Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany, August.

M. Stede. 2004. The Potsdam Commentary Corpus. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation*, pages 96–102, Barcelona, Spain, July.

P. Tapanainen and T. Järvinen. 1997. A nonprojective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington, DC, April.

K. Toutanova, D. Klein, C.D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003)*, Edmonton, Canada, May.

D. Tufiş. 2000. Using a large set of EAGLES-compliant morpho-syntactic descriptors as a tagset for probabilistic tagging. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, pages 1105–1112, Athens, Greece, May, 31st – June, 2nd.

A. Wagner and B. Zeisler. 2004. A syntactically annotated corpus of Tibetan. In *Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisboa, Portugal, May.

J. Zavrel and W. Daelemans. 2000. Bootstrapping a tagged corpus through combination of existing heterogeneous taggers. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece, May, 31st – June, 2nd.

A. Zielinski and C. Simon. 2008. Morphisto: An open-source morphological analyzer for German. In *Proceedings of the Conference on Finite State Methods in Natural Language Processing (FSMNLP)*, Ispra, Italy, September.

Temporal information processing of a new language: fast porting with minimal resources

Francisco Costa and **António Branco**
Universidade de Lisboa

Abstract

We describe the semi-automatic adaptation of a TimeML annotated corpus from English to Portuguese, a language for which TimeML annotated data was not available yet. In order to validate this adaptation, we use the obtained data to replicate some results in the literature that used the original English data. The fact that comparable results are obtained indicates that our approach can be used successfully to rapidly create semantically annotated resources for new languages.

1 Introduction

Temporal information processing is a topic of natural language processing boosted by recent evaluation campaigns like TERN2004,[1] TempEval-1 (Verhagen et al., 2007) and the forthcoming TempEval-2[2] (Pustejovsky and Verhagen, 2009). For instance, in the TempEval-1 competition, three tasks were proposed: a) identifying the temporal relation (such as *overlap*, *before* or *after*) holding between events and temporal entities such as dates, times and temporal durations denoted by expressions (i.e. temporal expressions) occurring in the same sentence; b) identifying the temporal relation holding between events expressed in a document and its creation time; c) identifying the temporal relation between the main events expressed by two adjacent sentences.

Supervised machine learning approaches are pervasive in the tasks of temporal information processing. Even when the best performing systems in these competitions are symbolic, there are machine learning solutions with results close to their performance. In TempEval-1, where there were statistical and rule-based systems, almost all systems achieved quite similar results. In the TERN2004 competition (aimed at identifying and normalizing temporal expressions), a symbolic system performed best, but since then machine learning solutions, such as (Ahn et al., 2007), have appeared that obtain similar results.

These evaluations made available sets of annotated data for English and other languages, used for training and evaluation. One natural question to ask is whether it is feasible to adapt the training and test data made available in these competitions to other languages, for which no such data still exist. Since the annotations are largely of a semantic nature, not many changes need to be done in the annotations once the textual material is translated. In essence, this would be a fast way to create temporal information processing systems for languages for which there are no annotated data yet.

In this paper, we report on an experiment that consisted in adapting the English data of TempEval-1 to Portuguese. The results of machine learning algorithms over the data thus obtained are compared to those reported for the English TempEval-1 competition. Since the results are quite similar, this permits to conclude that such an approach can rapidly generate relevant and comparable data and is useful when porting temporal information processing solutions to new languages.

The advantages of adapting an existing corpus instead of annotating text from scratch are: i) potentially less time consuming, if it is faster to translate the original text than it is to annotate new text (this can be the case if the annotations are semantic and complex); b) the annotations can be transposed without substantial modifications, which is the case if they are semantic in nature; c) less man power required: text annotation requires multiple annotators in order to guarantee the quality of the annotation tags, translation of the markables and transposition of the annotations

[1] http://timex2.mitre.org
[2] http://www.timeml.org/tempeval2

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 671–677,
Uppsala, Sweden, 11-16 July 2010.

in principle do not; d) the data obtained are comparable to the original data in all respects except for language: genre, domain, size, style, annotation decisions, etc., which allows for research to be conducted with a derived corpus that is comparable to research using the original corpus. There is of course the caveat that the adaptation process can introduce errors.

This paper proceeds as follows. In Section 2, we provide a quick overview of the TimeML annotations in the TempEval-1 data. In Section 3, it is described how the data were adapted to Portuguese. Section 4 contains a brief quantitative comparison of the two corpora. In Section 5, the results of replicating one of the approaches present in the TempEval-1 challenge with the Portuguese data are presented. We conclude this paper in Section 6.

2 Brief Description of the Annotations

Figure 1 contains an example of a document from the TempEval-1 corpus, which is similar to the TimeBank corpus (Pustejovsky et al., 2003).

In this corpus, event terms are tagged with `<EVENT>`. The relevant attributes are `tense`, `aspect`, `class`, `polarity`, `pos`, `stem`. The `stem` is the term's lemma, and `pos` is its part-of-speech. Grammatical tense and aspect are encoded in the features `tense` and `aspect`. The attribute `polarity` takes the value `NEG` if the event term is in a negative syntactic context, and `POS` otherwise. The attribute `class` contains several levels of information. It makes a distinction between terms that denote actions of speaking, which take the value `REPORTING` and those that do not. For these, it distinguishes between states (value `STATE`) and non-states (value `OCCURRENCE`), and it also encodes whether they create an intensional context (value `I_STATE` for states and value `I_ACTION` for non-states).

Temporal expressions (timexes) are inside `<TIMEX3>` elements. The most important features for these elements are `value`, `type` and `mod`. The timex's `value` encodes a normalized representation of this temporal entity, its `type` can be e.g. `DATE`, `TIME` or `DURATION`. The `mod` attribute is optional. It is used for expressions like *early this year*, which are annotated with `mod="START"`. As can be seen in Figure 1 there are other attributes for timexes that encode whether it is the document's creation time (`functionInDocument`) and whether its value can be determined from the expression alone or requires other sources of information (`temporalFunction` and `anchorTimeID`).

The `<TLINK>` elements encode temporal relations. The attribute `relType` represents the type of relation, the feature `eventID` is a reference to the first argument of the relation. The second argument is given by the attribute `relatedToTime` (if it is a time interval or duration) or `relatedToEvent` (if it is another event; this is for task C). The `task` feature is the name of the TempEval-1 task to which this temporal relation pertains.

3 Data Adaptation

We cleaned all TimeML markup in the TempEval-1 data and the result was fed to the Google Translator Toolkit.[3] This tool combines machine translation with a translation memory. A human translator corrected the proposed translations manually.

After that, we had the three collections of documents (the TimeML data, the English unannotated data and the Portuguese unannotated data) aligned by paragraphs (we just kept the line breaks from the original collection in the other collections). In this way, for each paragraph in the Portuguese data we know all the corresponding TimeML tags in the original English paragraph.

We tried using machine translation software (we used GIZA++ (Och and Ney, 2003)) to perform word alignment on the unannotated texts, which would have enabled us to transpose the TimeML annotations automatically. However, word alignment algorithms have suboptimal accuracy, so the results would have to be checked manually. Therefore we abandoned this idea, and instead we simply placed the different TimeML markup in the correct positions manually. This is possible since the TempEval-1 corpus is not very large. A small script was developed to place all relevant TimeML markup at the end of each paragraph in the Portuguese text, and then each tag was manually repositioned. Note that the `<TLINK>` elements always occur at the end of each document, each in a separate line: therefore they do not need to be repositioned.

During this manual repositioning of the annotations, some attributes were also changed man-

[3] http://translate.google.com/toolkit

```
<?xml version="1.0" ?>
<TempEval>

ABC<TIMEX3 tid="t52" type="DATE" value="1998-01-14" temporalFunction="false"
functionInDocument="CREATION_TIME">19980114</TIMEX3>.1830.0611
NEWS STORY

<s>In Washington <TIMEX3 tid="t53" type="DATE" value="1998-01-14" temporalFunction="true"
functionInDocument="NONE" anchorTimeID="t52">today</TIMEX3>, the Federal Aviation Administration <EVENT
eid="e1" class="OCCURRENCE" stem="release" aspect="NONE" tense="PAST" polarity="POS" pos="VERB">released
</EVENT> air traffic control tapes from <TIMEX3 tid="t54" type="TIME" value="1998-XX-XXTNI"
temporalFunction="true" functionInDocument="NONE" anchorTimeID="t52">the night</TIMEX3> the TWA Flight
eight hundred <EVENT eid="e2" class="OCCURRENCE" stem="go" aspect="NONE" tense="PAST" polarity="POS"
pos="VERB">went</EVENT>down.</s>
...
<TLINK lid="l1" relType="BEFORE" eventID="e2" relatedToTime="t53" task="A"/>
<TLINK lid="l2" relType="OVERLAP" eventID="e2" relatedToTime="t54" task="A"/>
<TLINK lid="l4" relType="BEFORE" eventID="e2" relatedToTime="t52" task="B"/>
...
</TempEval>
```

Figure 1: Extract of a document contained in the training data of the first TempEval-1

ually. In particular, the attributes `stem`, `tense` and `aspect` of `<EVENT>` elements are language specific and needed to be adapted. Sometimes, the `pos` attribute also needs to be changed, since e.g. a verb in English can be translated as a noun in Portuguese. The attribute `class` of the same kind of elements can be different, too, because natural sounding translations are sometimes not literal.

3.1 Annotation Decisions

When porting the TimeML annotations from English to Portuguese, a few decisions had to be made. For illustration purposes, Figure 2 contains the Portuguese equivalent of the extract presented in Figure 1.

For `<TIMEX3>` elements, the issue is that if the temporal expression to be annotated is a prepositional phrase, the preposition should not be inside the `<TIMEX3>` tags according to the TimeML specification. In the case of Portuguese, this raises the question of whether to leave contractions of prepositions with determiners outside these tags (in the English data the preposition is outside and the determiner is inside).[4] We chose to leave them outside, as can be seen in that Figure. In this example the prepositional phrase *from the night*/*da noite* is annotated with the English noun phrase *the night* inside the `<TIMEX3>` element, but the Portuguese version only contains the noun *noite* inside those tags.

For `<EVENT>` elements, some of the attributes are adapted. The value of the attribute `stem` is obviously different in Portuguese. The attributes `aspect` and `tense` have a different set of possible values in the Portuguese data, simply because the morphology of the two languages is different. In the example in Figure 1 the value `PPI` for the attribute `tense` stands for *pretérito perfeito do indicativo*. We chose to include mood information in the `tense` attribute because the different tenses of the indicative and the subjunctive moods do not line up perfectly as there are more tenses for the indicative than for the subjunctive. For the `aspect` attribute, which encodes grammatical aspect, we only use the values `NONE` and `PROGRESSIVE`, leaving out the values `PERFECTIVE` and `PERFECTIVE_PROGRESSIVE`, as in Portuguese there is no easy match between perfective aspect and grammatical categories.

The attributes of `<TIMEX3>` elements carry over to the Portuguese corpus unchanged, and the `<TLINK>` elements are taken verbatim from the original documents.

4 Data Description

The original English data for TempEval-1 are based on the TimeBank data, and they are split into one dataset for training and development and another dataset for evaluation. The full data are organized in 182 documents (162 documents in the training data and another 20 in the test data). Each document is a news report from television broadcasts or newspapers. A large amount of the documents (123 in the training set and 12 in the test data) are taken from a 1989 issue of the Wall Street Journal.

The training data comprise 162 documents with

[4] The fact that prepositions are placed outside of temporal expressions seems odd at first, but this is because in the original TimeBank, from which the TempEval data were derived, they are tagged as `<SIGNAL>`s. The TempEval-1 data does not contain `<SIGNAL>` elements, however.

```
<?xml version="1.0" encoding="UTF-8" ?>
<TempEval>

ABC<TIMEX3 tid="t52" type="DATE" value="1998-01-14" temporalFunction="false"
functionInDocument="CREATION_TIME">19980114</TIMEX3>.1830.1611
REPORTAGEM

<s>Em Washington, <TIMEX3 tid="t53" type="DATE" value="1998-01-14" temporalFunction="true"
functionInDocument="NONE" anchorTimeID="t52">hoje</TIMEX3>, a Federal Aviation Administration <EVENT
eid="e1" class="OCCURRENCE" stem="publicar" aspect="NONE" tense="PPI" polarity="POS" pos="VERB">publicou
</EVENT> gravaoes do controlo de trfego areo da <TIMEX3 tid="t54" type="TIME" value="1998-XX-XXTNI"
temporalFunction="true" functionInDocument="NONE" anchorTimeID="t52">noite</TIMEX3> em que o voo TWA800
<EVENT eid="e2" class="OCCURRENCE" stem="cair" aspect="NONE" tense="PPI" polarity="POS" pos="VERB">caiu
</EVENT>
.</s>
...
<TLINK lid="l1" relType="BEFORE" eventID="e2" relatedToTime="t53" task="A"/>
<TLINK lid="l2" relType="OVERLAP" eventID="e2" relatedToTime="t54" task="A"/>
<TLINK lid="l4" relType="BEFORE" eventID="e2" relatedToTime="t52" task="B"/>
...
</TempEval>
```

Figure 2: Extract of a document contained in the Portuguese data

2,236 sentences (i.e. 2236 <s> elements) and 52,740 words. It contains 6799 <EVENT> elements, 1,244 <TIMEX3> elements and 5,790 <TLINK> elements. Note that not all the events are included here: the ones expressed by words that occur less than 20 times in TimeBank were removed from the TempEval-1 data.

The test dataset contains 376 sentences and 8,107 words. The number of <EVENT> elements is 1,103; there are 165 <TIMEX3>s and 758 <TLINK>s.

The Portuguese data of course contain the same (translated) documents. The training dataset has 2,280 sentences and 60,781 words. The test data contains 351 sentences and 8,920 words.

5 Comparing the two Datasets

One of the systems participating in the TempEval-1 competition, the USFD system (Hepple et al., 2007), implemented a very straightforward solution: it simply trained classifiers with Weka (Witten and Frank, 2005), using as attributes information that was readily available in the data and did not require any extra natural language processing (for all tasks, the attribute relType of <TLINK> elements is unknown and must be discovered, but all the other information is given).

The authors' objectives were to see "whether a 'lite' approach of this kind could yield reasonable performance, before pursuing possibilities that relied on 'deeper' NLP analysis methods", "which of the features would contribute positively to system performance" and "if any [machine learning] approach was better suited to the TempEval tasks than any other". In spite of its simplicity, they obtained results quite close to the best systems.

For us, the results of (Hepple et al., 2007) are interesting as they allow for a straightforward evaluation of our adaptation efforts, since the same machine learning implementations can be used with the Portuguese data, and then compared to their results.

The differences in the data are mostly due to language. Since the languages are different, the distribution of the values of several attributes are different. For instance, we included both tense and mood information in the tense attribute of <EVENT>s, as mentioned in Section 3.1, so instead of seven possible values for this attribute, the Portuguese data contains more values, which can cause more data sparseness. Other attributes affected by language differences are aspect, pos, and class, which were also possibly changed during the adaptation process.

One important difference between the English and the Portuguese data originates from the fact that events with a frequency lower than 20 were removed from the English TempEval-1 data. Since there is not a 1 to 1 relation between English event terms and Portuguese event terms, we do not have the guarantee that all event terms in the Portuguese data have a frequency of at least 20 occurrences in the entire corpus.[5]

The work of (Hepple et al., 2007) reports on both cross-validation results for various classifiers over the training data and evaluation results on the training data, for the English dataset. We we will

[5] In fact, out of 1,649 different stems for event terms in the Portuguese training data, only 45 occur at least 20 times.

	Task		
Attribute	A	B	C
EVENT-aspect	✓	✓	✓
EVENT-polarity	✓	✓	×
EVENT-POS	✓	✓	✓
EVENT-stem	✓	×	×
EVENT-string	×	×	×
EVENT-class	×	✓	✓
EVENT-tense	×	✓	✓
ORDER-adjacent	✓	N/A	N/A
ORDER-event-first	✓	N/A	N/A
ORDER-event-between	×	N/A	N/A
ORDER-timex-between	×	N/A	N/A
TIMEX3-mod	✓	×	N/A
TIMEX3-type	✓	×	N/A

Table 1: Features used for the English TempEval-1 tasks. N/A means the feature was not applicable to the task, ✓ means the feature was used by the best performing classifier for the task, and × means it was not used by that classifier. From (Hepple et al., 2007).

be comparing their results to ours.

Our purpose with this comparison is to validate the corpus adaptation. Similar results would not necessarily indicate the quality of the adapted corpus. After all, a word-by-word translation would produce data that would yield similar results, but it would also be a very poor translation, and therefore the resulting corpus would not be very interesting. The quality of the translation is not at stake here, since it was manually revised. But similar results would indicate that the obtained data are comparable to the original data, and that they are similarly useful to tackle the problem for which the original data were collected. This would confirm our hypothesis that adapting an existing corpus can be an effective way to obtain new data for a different language.

5.1 Results for English

The attributes employed for English by (Hepple et al., 2007) are summarized in Table 1. The class is the attribute `relType` of `<TLINK>` elements.

The EVENT features are taken from `<EVENT>` elements. The EVENT-`string` attribute is the character data inside the element. The other attributes correspond to the feature of `<EVENT>` with the same name. The TIMEX3 features also correspond to attributes of the relevant `<TIMEX3>` element. The ORDER features are boolean and computed as follows:

- ORDER-`event-first` is whether the `<EVENT>` element occurs in the text before the `<TIMEX3>` element;
- ORDER-`event-between` is whether an `<EVENT>` element occurs in the text between the two temporal entities being ordered;
- ORDER-`timex-between` is the same, but for temporal expressions;
- ORDER-`adjacent` is whether both ORDER-`event-between` and ORDER-`timex-between` are false (but other textual data may occur between the two entities).

	Task		
Algorithm	A	B	C
baseline	49.8	62.1	42.0
`lazy.KStar`	**58.2**	76.7	54.0
`rules.DecisionTable`	53.3	**79.0**	52.9
`functions.SMO`	55.1	78.1	**55.5**
`rules.JRip`	50.7	78.6	53.4
`bayes.NaiveBayes`	56.3	76.2	50.7

Table 2: Performance of several machine learning algorithms on the English TempEval-1 training data, with cross-validation. The best result for each task is in boldface. From (Hepple et al., 2007).

Cross-validation over the training data produced the results in Table 2. The baseline used is the majority class baseline, as given by Weka's `rules.ZeroR` implementation. The `lazy.KStar` algorithm is a nearest-neighbor classifier that uses an entropy-based measure to compute instance similarity. Weka's `rules.DecisionTable` algorithm assigns to an unknown instance the majority class of the training examples that have the same attribute values as that instance that is being classified. `functions.SMO` is an implementation of Support Vector Machines (SVM), `rules.JRip` is the RIPPER algorithm, and `bayes.NaiveBayes` is a Naive Bayes classifier.

Algorithm	Task A	B	C
baseline	49.8	62.1	42.0
`lazy.KStar`	**57.4**	77.7	53.3
`rules.DecisionTable`	54.2	78.1	51.6
`functions.SMO`	55.5	**79.3**	56.8
`rules.JRip`	52.1	77.6	52.1
`bayes.NaiveBayes`	56.0	78.2	53.5
`trees.J48`	55.6	79.0	**59.3**

Table 3: Performance of several machine learning algorithms on the Portuguese data for the TempEval-1 tasks. The best result for each task is in boldface.

5.2 Attributes

We created a small script to convert the XML annotated files into CSV files, that can be read by Weka. In this process, we included the same attributes as the USFD authors used for English.

For task C, (Hepple et al., 2007) are not very clear whether the EVENT attributes used were related to just one of the two events being temporally related. In any case, we used two of each of the EVENT attributes, one for each event in the temporal relation to be determined. So, for instance, an extra attribute EVENT2-`tense` is where the tense of the second event in the temporal relation is kept.

5.3 Results

The majority class baselines produce the same results as for English. This was expected: the class distribution is the same in the two datasets, since the `<TLINK>` elements were copied to the adapted corpus without any changes.

For the sake of comparison, we used the same classifiers as (Hepple et al., 2007), and we used the attributes that they found to work best for English (presented above in Table 1). The results for the Portuguese dataset are in Table 3, using 10-fold cross-validation on the training data.

We also present the results for Weka's implementation of the C4.5 algorithm, to induce decision trees. The motivation to run this algorithm over these data is that decision trees are human readable and make it easy to inspect what decisions the classifier is making. This is also true of `rules.JRip`. The results for the decision trees are in this table, too.

The results obtained are almost identical to the results for the original dataset in English. The best performing classifier for task A is the same as for English. For task B, Weka's `functions.SMO` produced better results with the Portuguese data than `rules.DecisionTable`, the best performing classifier with the English data for this task. In task C, the SVM algorithm was also the best performing algorithm among those that were also tried on the English data, but decision trees produced even better results here.

For English, the best performing classifier for each task on the training data, according to Table 2, was used for evaluation on the test data: the results showed a 59% F-measure for task A, 73% for task B, and 54% for task C.

Similarly, we also evaluated the best algorithm for each task (according to Table 3) with the Portuguese test data, after training it on the entire training dataset. The results are: in task A the `lazy.KStar` classifier scored 58.6%, and the SVM classifier scored 75.5% in task B and 59.4% in task C, with `trees.J48` scoring 61% in this task.

The results on the test data are also fairly similar for the two languages/datasets.

We inspected the decision trees and rule sets produced by `trees.J48` and `rules.JRip`, in order to see what the classifiers are doing.

Task B is probably the easiest task to check this way, because we expect grammatical tense to be highly predictive of the temporal order between an event and the document's creation time.

And, indeed, the top of the tree induced by `trees.J48` is quite interesting:

```
eTense =  PI:  OVERLAP (388.0/95.0)
eTense =  PPI:  BEFORE (1051.0/41.0)
```

Here, `eTense` is the EVENT-`tense` attribute of `<EVENT>` elements, `PI` stands for present indicative, and `PPI` is past indicative (*pretérito perfeito do indicativo*). In general, one sees past tenses associated with the `BEFORE` class and future tenses associated with the `AFTER` class (including the conditional forms of verbs). Infinitives are mostly associated with the `AFTER` class, and present subjunctive forms with `AFTER` and `OVERLAP`. Figure 3 shows the rule set induced by the RIPPER algorithm.

The classifiers for the other tasks are more difficult to inspect. For instance, in task A, the event term and the temporal expression that denote the entities that are to be ordered may not even be directly syntactically related. Therefore, it is hard to

```
(eClass = OCCURRENCE) and ( eTense =  INF) and ( ePolarity =  POS) =>  lRelType= AFTER
                                                                          (183.0/77.0)
( eTense =  FI) =>  lRelType= AFTER (55.0/10.0)
(eClass = OCCURRENCE) and ( eTense =  IR-PI+INF) =>  lRelType= AFTER (26.0/4.0)
(eClass = OCCURRENCE) and ( eTense =  PC) =>  lRelType= AFTER (15.0/3.0)
(eClass = OCCURRENCE) and ( eTense =  C) =>  lRelType= AFTER (17.0/2.0)
( eTense =  PI) =>  lRelType= OVERLAP (388.0/95.0)
(eClass = ASPECTUAL) and ( eTense =  PC) =>  lRelType= OVERLAP (9.0/2.0)
 =>  lRelType= BEFORE (1863.0/373.0)
```

Figure 3: `rules.JRip` classifier induced for task B. `INF` stands for infinitive, `FI` is future indicative, `IR-PI+INF` is an infinitive form following a present indicative form of the verb *ir* (*to go*), `PC` is present subjunctive, `C` is conditional, `PI` is present indicative.

see how interesting the inferred rules are, because we do not know what would be interesting in this scenario. In any case, the top of the induced tree for task A is:

```
oAdjacent =  True: OVERLAP (554.0/128.0)
```

Here, `oAdjacent` is the ORDER-`adjacent` attribute. Assuming this attribute is an indication that the event term and the temporal expression are related syntactically, it is interesting to see that the typical temporal relation between the two entities in this case is an `OVERLAP` relation. The rest of the tree is much more *ad-hoc*, making frequent use of the `stem` attribute of `<EVENT>` elements, suggesting the classifier is memorizing the data.

Task C, where two events are to be ordered, produced more complicated classifiers. Generally the induced rules and the tree paths compare the tense and the class of the two event terms, showing some expected heuristics (such as, if the tense of the first event is future and the tense of the second event is past, assign `AFTER`). But there are also many several rules for which we do not have clear intuitions.

6 Discussion

In this paper, we described the semi-automatic adaptation of a TimeML annotated corpus from English to Portuguese, a language for which TimeML annotated data was not available yet.

Because most of the TimeML annotations are semantic in nature, they can be transposed to a translation of the original corpus, with few adaptations being required.

In order to validate this adaptation, we used the obtained data to replicate some results in the literature that used the original English data.

The results for the Portuguese data are very similar to the ones for English. This indicates that our approach to adapt existing annotated data to a different language is fruitful.

References

David Ahn, Joris van Rantwijk, and Maarten de Rijke. 2007. A cascaded machine learning approach to interpreting temporal expressions. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 420–427, Rochester, New York, April. Association for Computational Linguistics.

Mark Hepple, Andrea Setzer, and Rob Gaizauskas. 2007. USFD: Preliminary exploration of features and classifiers for the TempEval-2007 tasks. In *Proceedings of SemEval-2007*, pages 484–487, Prague, Czech Republic. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

James Pustejovsky and Marc Verhagen. 2009. Semeval-2010 task 13: evaluating events, time expressions, and temporal relations (tempeval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 112–116, Boulder, Colorado. Association for Computational Linguistics.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, and J. Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *Proceedings of SemEval-2007*.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco. second edition.

A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation

Stephen Tratz and **Eduard Hovy**
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{stratz,hovy}@isi.edu

Abstract

The automatic interpretation of noun-noun compounds is an important subproblem within many natural language processing applications and is an area of increasing interest. The problem is difficult, with disagreement regarding the number and nature of the relations, low inter-annotator agreement, and limited annotated data. In this paper, we present a novel taxonomy of relations that integrates previous relations, the largest publicly-available annotated dataset, and a supervised classification method for automatic noun compound interpretation.

1 Introduction

Noun compounds (e.g., 'maple leaf') occur very frequently in text, and their interpretation—determining the relationships between adjacent nouns as well as the hierarchical dependency structure of the NP in which they occur—is an important problem within a wide variety of natural language processing (NLP) applications, including machine translation (Baldwin and Tanaka, 2004) and question answering (Ahn et al., 2005). The interpretation of noun compounds is a difficult problem for various reasons (Spärck Jones, 1983). Among them is the fact that no set of relations proposed to date has been accepted as complete and appropriate for general-purpose text. Regardless, automatic noun compound interpretation is the focus of an upcoming SEMEVAL task (Butnariu et al., 2009).

Leaving aside the problem of determining the dependency structure among strings of three or more nouns—a problem we do not address in this paper—automatic noun compound interpretation requires a taxonomy of noun-noun relations, an automatic method for accurately assigning the relations to noun compounds, and, in the case of supervised classification, a sufficiently large dataset for training.

Earlier work has often suffered from using taxonomies with coarse-grained, highly ambiguous predicates, such as prepositions, as various labels (Lauer, 1995) and/or unimpressive inter-annotator agreement among human judges (Kim and Baldwin, 2005). In addition, the datasets annotated according to these various schemes have often been too small to provide wide coverage of the noun compounds likely to occur in general text.

In this paper, we present a large, fine-grained taxonomy of 43 noun compound relations, a dataset annotated according to this taxonomy, and a supervised, automatic classification method for determining the relation between the head and modifier words in a noun compound. We compare and map our relations to those in other taxonomies and report the promising results of an inter-annotator agreement study as well as an automatic classification experiment. We examine the various features used for classification and identify one very useful, novel family of features. Our dataset is, to the best of our knowledge, the largest noun compound dataset yet produced. We will make it available via http://www.isi.edu.

2 Related Work

2.1 Taxonomies

The relations between the component nouns in noun compounds have been the subject of various linguistic studies performed throughout the years, including early work by Jespersen (1949). The taxonomies they created are varied. Lees created an early taxonomy based primarily upon grammar (Lees, 1960). Levi's influential work postulated that *complex nominals* (Levi's name for noun compounds that also permits certain adjectival modifiers) are all derived either via nominalization or

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 678–687,
Uppsala, Sweden, 11-16 July 2010.

by deleting one of nine predicates (i.e., CAUSE, HAVE, MAKE, USE, BE, IN, FOR, FROM, ABOUT) from an underlying sentence construction (Levi, 1978). Of the taxonomies presented by purely linguistic studies, our categories are most similar to those proposed by Warren (1978), whose categories (e.g., MATERIAL+ARTEFACT, OBJ+PART) are generally less ambiguous than Levi's.

In contrast to studies that claim the existence of a relatively small number of semantic relations, Downing (1977) presents a strong case for the existence of an unbounded number of relations. While we agree with Downing's belief that the number of relations is unbounded, we contend that the vast majority of noun compounds fits within a relatively small set of categories.

The relations used in computational linguistics vary much along the same lines as those proposed earlier by linguists. Several lines of work (Finin, 1980; Butnariu and Veale, 2008; Nakov, 2008) assume the existence of an unbounded number of relations. Others use categories similar to Levi's, such as Lauer's (1995) set of prepositional paraphrases (i.e., OF, FOR, IN, ON, AT, FROM, WITH, ABOUT) to analyze noun compounds. Some work (e.g., Barker and Szpakowicz, 1998; Nastase and Szpakowicz, 2003; Girju et al., 2005; Kim and Baldwin, 2005) use sets of categories that are somewhat more similar to those proposed by Warren (1978). While most of the noun compound research to date is not domain specific, Rosario and Hearst (2001) create and experiment with a taxonomy tailored to biomedical text.

2.2 Classification

The approaches used for automatic classification are also varied. Vanderwende (1994) presents one of the first systems for automatic classification, which extracted information from online sources and used a series of rules to rank a set of most likely interpretations. Lauer (1995) uses corpus statistics to select a prepositional paraphrase. Several lines of work, including that of Barker and Szpakowicz (1998), use memory-based methods. Kim and Baldwin (2005) and Turney (2006) use nearest neighbor approaches based upon WordNet (Fellbaum, 1998) and Turney's Latent Relational Analysis, respectively. Rosario and Hearst (2001) utilize neural networks to classify compounds according to their domain-specific relation taxonomy. Moldovan et al. (2004) use SVMs as well as a novel algorithm (i.e., semantic scattering). Nastase et al. (2006) experiment with a variety of classification methods including memory-based methods, SVMs, and decision trees. Ó Séaghdha and Copestake (2009) use SVMs and experiment with kernel methods on a dataset labeled using a relatively small taxonomy. Girju (2009) uses cross-linguistic information from parallel corpora to aid classification.

3 Taxonomy

3.1 Creation

Given the heterogeneity of past work, we decided to start fresh and build a new taxonomy of relations using naturally occurring noun pairs, and then compare the result to earlier relation sets. We collected 17509 noun pairs and over a period of 10 months assigned one or more relations to each, gradually building and refining our taxonomy. More details regarding the dataset are provided in Section 4.

The relations we produced were then compared to those present in other taxonomies (e.g., Levi, 1978; Warren, 1978; Barker and Szpakowicz, 1998; Girju et al., 2005), and they were found to be fairly similar. We present a detailed comparison in Section 3.4.

We tested the relation set with an initial inter-annotator agreement study (our latest inter-annotator agreement study results are presented in Section 6). However, the mediocre results indicated that the categories and/or their definitions needed refinement. We then embarked on a series of changes, testing each generation by annotation using Amazon's Mechanical Turk service, a relatively quick and inexpensive online platform where requesters may publish tasks for anonymous online workers (Turkers) to perform. Mechanical Turk has been previously used in a variety of NLP research, including recent work on noun compounds by Nakov (2008) to collect short phrases for linking the nouns within noun compounds.

For the Mechanical Turk annotation tests, we created five sets of 100 noun compounds from noun compounds automatically extracted from a random subset of New York Times articles written between 1987 and 2007 (Sandhaus, 2008). Each of these sets was used in a separate annotation round. For each round, a set of 100 noun compounds was uploaded along with category defini-

Category Name	% Example	Approximate Mappings
Causal Group		
COMMUNICATOR OF COMMUNICATION	0.77 court order	$\supset$BGN:Agent, $\supset$L:Act$_a$+Product$_a$, $\supset$V:Subj
PERFORMER OF ACT/ACTIVITY	2.07 police abuse	$\supset$BGN:Agent, $\supset$L:Act$_a$+Product$_a$, $\supset$V:Subj
CREATOR/PROVIDER/CAUSE OF	2.55 ad revenue	$\subset$BGV:Cause(d-by), $\subset$L:Cause$_2$, $\subset$N:Effect
Purpose/Activity Group		
PERFORM/ENGAGE_IN	13.24 cooking pot	$\supset$BGV:Purpose, $\supset$L:For, $\approx$N:Purpose, $\supset$W:Activity$\cup$Purpose
CREATE/PROVIDE/SELL	8.94 nicotine patch	∞BV:Purpose, $\subset$BG:Result, ∞G:Make-Produce, $\subset$GNV:Cause(s), ∞L:Cause$_1\cup$Make$_1\cup$For, $\subset$N:Product, $\supset$W:Activity$\cup$Purpose
OBTAIN/ACCESS/SEEK	1.50 shrimp boat	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
MODIFY/PROCESS/CHANGE	1.50 eye surgery	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
MITIGATE/OPPOSE/DESTROY	2.34 flak jacket	$\supset$BGV:Purpose, $\supset$L:For, $\approx$N:Detraction, $\supset$W:Activity$\cup$Purpose
ORGANIZE/SUPERVISE/AUTHORITY	4.82 ethics board	$\supset$BGNV:Purpose/Topic, $\supset$L:For/About$_a$, $\supset$W:Activity
PROPEL	0.16 water gun	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
PROTECT/CONSERVE	0.25 screen saver	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
TRANSPORT/TRANSFER/TRADE	1.92 freight train	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
TRAVERSE/VISIT	0.11 tree traversal	$\supset$BGNV:Purpose, $\supset$L:For, $\supset$W:Activity$\cup$Purpose
Ownership, Experience, Employment, and Use		
POSSESSOR + OWNED/POSSESSED	2.11 family estate	$\supset$BGNVW:Possess*, $\supset$L:Have$_2$
EXPERIENCER + COGINITION/MENTAL	0.45 voter concern	$\supset$BNVW:Possess*, $\approx$G:Experiencer, $\supset$L:Have$_2$
EMPLOYER + EMPLOYEE/VOLUNTEER	2.72 team doctor	$\supset$BGNVW:Possess*, $\supset$L:For/Have$_2$, $\supset$BGN:Beneficiary
CONSUMER + CONSUMED	0.09 cat food	$\supset$BGNVW:Purpose, $\supset$L:For, $\supset$BGN:Beneficiary
USER/RECIPIENT + USED/RECEIVED	1.02 voter guide	$\supset$BNVW:Purpose, $\supset$G:Recipient, $\supset$L:For, $\supset$BGN:Beneficiary
OWNED/POSSESSED + POSSESSION	1.20 store owner	$\approx$G:Possession, $\supset$L:Have$_1$, $\approx$W:Belonging-Possessor
EXPERIENCE + EXPERIENCER	0.27 fire victim	$\approx$G:Experiencer, ∞L:Have$_1$
THING CONSUMED + CONSUMER	0.41 fruit fly	$\supset$W:Obj-SingleBeing
THING/MEANS USED + USER	1.96 faith healer	$\approx$BNV:Instrument, $\approx$G:Means$\cup$Instrument, $\approx$L:Use, $\subset$W:MotivePower-Obj
Temporal Group		
TIME [SPAN] + X	2.35 night work	$\approx$BNV:Time(At), $\supset$G:Temporal, $\approx$L:In$_c$, $\approx$W:Time-Obj
X + TIME [SPAN]	0.50 birth date	$\supset$G:Temporal, $\approx$W:Obj-Time
Location and Whole+Part/Member of		
LOCATION/GEOGRAPHIC SCOPE OF X	4.99 hillside home	$\approx$BGV:Locat(ion/ive), $\approx$L:In$_a\cup$From$_b$, B:Source, $\approx$N:Location(At/From), $\approx$W:Place-Obj$\cup$PlaceOfOrigin
WHOLE + PART/MEMBER OF	1.75 robot arm	$\supset$B:Possess*, $\approx$G:Part-Whole, $\supset$L:Have$_2$, $\approx$N:Part, $\approx$V:Whole-Part, $\approx$W:Obj-Part$\cup$Group-Member
Composition and Containment Group		
SUBSTANCE/MATERIAL/INGREDIENT + WHOLE	2.42 plastic bag	$\subset$BNVW:Material*, ∞GN:Source, ∞L:From$_a$, $\approx$L:Have$_1$, ∞L:Make$_{2b}$, ∞N:Content
PART/MEMBER + COLLECTION/CONFIG/SERIES	1.78 truck convoy	$\approx$L:Make$_{2ac}$, $\approx$N:Whole, $\approx$V:Part-Whole, $\approx$W:Parts-Whole
X + SPATIAL CONTAINER/LOCATION/BOUNDS	1.39 shoe box	$\supset$B:Content$\cup$Located, $\supset$L:For, $\supset$L:Have$_1$, $\approx$N:Location, $\approx$W:Obj-Place
Topic Group		
TOPIC OF COMMUNICATION/IMAGERY/INFO	8.37 travel story	$\supset$BGNV:Topic, $\supset$L:About$_{ab}$, $\supset$W:SubjectMatter, $\subset$G:Depiction
TOPIC OF PLAN/DEAL/ARRANGEMENT/RULES	4.11 loan terms	$\supset$BGNV:Topic, $\supset$L:About$_a$, $\supset$W:SubjectMatter
TOPIC OF OBSERVATION/STUDY/EVALUATION	1.71 job survey	$\supset$BGNV:Topic, $\supset$L:About$_a$, $\supset$W:SubjectMatter
TOPIC OF COGNITION/EMOTION	0.58 jazz fan	$\supset$BGNV:Topic, $\supset$L:About$_a$, $\supset$W:SubjectMatter
TOPIC OF EXPERT	0.57 policy wonk	$\supset$BGNV:Topic, $\supset$L:About$_a$, $\supset$W:SubjectMatter
TOPIC OF SITUATION	1.64 oil glut	$\supset$BGNV:Topic, $\approx$L:About$_c$
TOPIC OF EVENT/PROCESS	1.09 lava flow	$\supset$G:Theme, $\supset$V:Subj
Attribute Group		
TOPIC/THING + ATTRIB	4.13 street name	$\supset$BNV:Possess*, $\approx$G:Property, $\supset$L:Have$_2$, $\approx$W:Obj-Quality
TOPIC/THING + ATTRIB VALUE CHARAC OF	0.31 earth tone	
Attributive and Coreferential		
COREFERENTIAL	4.51 fighter plane	$\approx$BV:Equative, $\supset$G:Type$\cup$IS-A, $\approx$L:BE$_{bcd}$, $\approx$N:Type$\cup$Equality, $\approx$W:Copula
PARTIAL ATTRIBUTE TRANSFER	0.69 skeleton crew	$\approx$W:Resemblance, $\supset$G:Type
MEASURE + WHOLE	4.37 hour meeting	$\approx$G:Measure, $\subset$N:TimeThrough$\cup$Measure, $\approx$W:Size-Whole
Other		
HIGHLY LEXICALIZED / FIXED PAIR	0.65 pig iron	
OTHER	1.67 contact lens	

Table 1: The semantic relations, their frequency in the dataset, examples, and *approximate* relation mappings to previous relation sets. $\approx$-approximately equivalent; $\supset$/$\subset$-super/sub set; ∞-some overlap; $\cup$-union; initials BGLNVW refer respectively to the works of (Barker and Szpakowicz, 1998; Girju et al., 2005; Girju, 2007; Levi, 1978; Nastase and Szpakowicz, 2003; Vanderwende, 1994; Warren, 1978).

tions and examples. Turkers were asked to select one or, if they deemed it appropriate, two categories for each noun pair. After all annotations for the round were completed, they were examined, and any taxonomic changes deemed appropriate (e.g., the creation, deletion, and/or modification of categories) were incorporated into the taxonomy before the next set of 100 was uploaded. The categories were substantially modified during this process. They are shown in Table 1 along with examples and an approximate mapping to several other taxonomies.

3.2 Category Descriptions

Our categories are defined with sentences. For example, the SUBSTANCE category has the definition *n_1 is one of the primary physical substances/materials/ingredients that n_2 is made/composed out of/from*. Our LOCATION category's definition reads *n_1 is the location / geographic scope where n_2 is at, near, from, generally found, or occurs*. Defining the categories with sentences is advantageous because it is possible to create straightforward, explicit defintions that humans can easily test examples against.

3.3 Taxonomy Groupings

In addition to influencing the category definitions, some taxonomy groupings were altered with the hope that this would improve inter-annotator agreement for cases where Turker disagreement was systematic. For example, LOCATION and WHOLE + PART/MEMBER OF were commonly disagreed upon by Turkers so they were placed within their own taxonomic subgroup. The ambiguity between these categories has previously been observed by Girju (2009).

Turkers also tended to disagree between the categories related to composition and containment. Due this apparent similarity they were also grouped together in the taxonomy.

The ATTRIBUTE categories are positioned near the TOPIC group because some Turkers chose a TOPIC category when an ATTRIBUTE category was deemed more appropriate. This may be because attributes are relatively abstract concepts that are often somewhat *descriptive* of whatever possesses them. A prime example of this is *street name*.

3.4 Contrast with other Taxonomies

In order to ensure completeness, we mapped into our taxonomy the relations proposed in most previous work including those of Barker and Szpakowicz (1998) and Girju et al. (2005). The results, shown in Table 1, demonstrate that our taxonomy is similar to several taxonomies used in other work. However, there are three main differences and several less important ones. The first major difference is the absence of a significant THEME or OBJECT category. The second main difference is that our taxonomy does not include a PURPOSE category and, instead, has several smaller categories. Finally, instead of possessing a single TOPIC category, our taxonomy has several, finer-grained TOPIC categories. These differences are significant because THEME/OBJECT, PURPOSE, and TOPIC are typically among the most frequent categories.

THEME/OBJECT is typically the category to which other researchers assign noun compounds whose head noun is a nominalized verb and whose modifier noun is the THEME/OBJECT of the verb. This is typically done with the justification that the relation/predicate (the root verb of the nominalization) is overtly expressed.

While including a THEME/OBJECT category has the advantage of simplicity, its disadvantages are significant. This category leads to a significant ambiguity in examples because many compounds fitting the THEME/OBJECT category also match some other category as well. Warren (1978) gives the examples of *soup pot* and *soup container* to illustrate this issue, and Girju (2009) notes a substantial overlap between THEME and MAKE-PRODUCE. Our results from Mechanical Turk showed significant overlap between PURPOSE and OBJECT categories (present in an earlier version of the taxonomy). For this reason, we do not include a separate THEME/OBJECT category. If it is important to know whether the modifier also holds a THEME/OBJECT relationship, we suggest treating this as a separate classification task.

The absence of a single PURPOSE category is another distinguishing characteristic of our taxonomy. Instead, the taxonomy includes a number of finer-grained categories (e.g., PERFORM/ENGAGE_IN), which can be conflated to create a PURPOSE category if necessary. During our Mechanical Turk-based refinement process, our now-defunct PURPOSE category was found to be ambiguous with many other categories as well as difficult to define. This problem has been noted by others. For example, Warren (1978)

points out that *tea* in *tea cup* qualifies as both the *content* and the *purpose* of the *cup*. Similarly, while WHOLE+PART/MEMBER was selected by most Turkers for *bike tire*, one individual chose PURPOSE. Our investigation identified five main purpose-like relations that most of our PURPOSE examples can be divided into, including activity performance (PERFORM/ENGAGE_IN), creation/provision (CREATE/PROVIDE/CAUSE OF), obtainment/access (OBTAIN/ACCESS/SEEK), supervision/management (ORGANIZE/SUPERVISE/AUTHORITY), and opposition (MITIGATE/OPPOSE/DESTROY).

The third major distinguishing different between our taxonomy and others is the absence of a single TOPIC/ABOUT relation. Instead, our taxonomy has several finer-grained categories that can be conflated into a TOPIC category. Unlike the previous two distinguishing characteristics, which were motivated primarily by Turker annotations, this separation was largely motivated by author dissatisfaction with a single TOPIC category.

Two differentiating characteristics of less importance are the absence of BENEFICIARY or SOURCE categories (Barker and Szpakowicz, 1998; Nastase and Szpakowicz, 2003; Girju et al., 2005). Our EMPLOYER, CONSUMER, and USER/RECIPIENT categories combined more or less cover BENEFICIARY. Since SOURCE is ambiguous in multiple ways including causation (*tsunami injury*), provision (*government grant*), ingredients (*rice wine*), and locations (*north wind*), we chose to exclude it.

4 Dataset

Our noun compound dataset was created from two principal sources: an in-house collection of terms extracted from a large corpus using part-of-speech tagging and mutual information and the Wall Street Journal section of the Penn Treebank. Compounds including one or more proper nouns were ignored. In total, the dataset contains 17509 unique, out-of-context examples, making it by far the largest hand-annotated compound noun dataset in existence that we are aware of. Proper nouns were not included.

The next largest available datasets have a variety of drawbacks for noun compound interpretation in general text. Kim and Baldwin's (2005) dataset is the second largest available dataset, but inter-annotator agreement was only 52.3%, and the annotations had an usually lopsided distribution; 42% of the data has TOPIC labels. Most (73.23%) of Girju's (2007) dataset consists of noun-preposition-noun constructions. Rosario and Heart's (2001) dataset is specific to the biomedical domain, while Ó Séaghdha and Copestake's (2009) data is labeled with only 5 extremely coarse-grained categories. The remaining datasets are too small to provide wide coverage. See Table 2 below for size comparison with other publicly available, semantically annotated datasets.

Size	Work
17509	Tratz and Hovy, 2010
2169	Kim and Baldwin, 2005
2031	*Girju, 2007*
1660	Rosario and Hearst, 2001
1443	Ó Séaghdha and Copestake, 2007
505	*Barker and Szpakowicz, 1998*
600	*Nastase and Szpakowicz, 2003*
395	Vanderwende, 1994
385	Lauer, 1995

Table 2: Size of various available noun compound datasets labeled with relation annotations. Italics indicate that the dataset contains n-prep-n constructions and/or non-nouns.

5 Automated Classification

We use a Maximum Entropy (Berger et al., 1996) classifier with a large number of boolean features, some of which are novel (e.g., the inclusion of words from WordNet definitions). Maximum Entropy classifiers have been effective on a variety of NLP problems including preposition sense disambiguation (Ye and Baldwin, 2007), which is somewhat similar to noun compound interpretation. We use the implementation provided in the MALLET machine learning toolkit (McCallum, 2002).

5.1 Features Used

WordNet-based Features

- {Synonyms, Hypernyms} for all NN and VB entries for each word
- Intersection of the words' hypernyms
- All terms from the 'gloss' for each word
- Intersection of the words' 'gloss' terms
- Lexicographer file names for each word's NN and VB entries (e.g., n_1:substance)

- Logical AND of lexicographer file names for the two words (e.g., n_1:substance $\wedge$ n_2:artifact)
- Lists of all link types (e.g., meronym links) associated with each word
- Logical AND of the link types (e.g., n_1:hasMeronym(s) $\wedge$ n_2:hasHolonym(s))
- Part-of-speech (POS) indicators for the existence of VB, ADJ, and ADV entries for each of the nouns
- Logical AND of the POS indicators for the two words
- 'Lexicalized' indicator for the existence of an entry for the compound as a single term
- Indicators if either word is a part of the other word according to Part-Of links
- Indicators if either word is a hypernym of the other
- Indicators if either word is in the definition of the other

Roget's Thesaurus-based Features

- Roget's divisions for all noun (and verb) entries for each word
- Roget's divisions shared by the two words

Surface-level Features

- Indicators for the suffix types (e.g., de-adjectival, de-nominal [non]agentive, de-verbal [non]agentive)
- Indicators for degree, number, order, or locative prefixes (e.g., ultra-, poly-, post-, and inter-, respectively)
- Indicators for whether or not a preposition occurs within either term (e.g., 'down' in 'breakdown')
- The last {two, three} letters of each word

Web 1T N-gram Features

To provide information related to term usage to the classifier, we extracted trigram and 4-gram features from the Web 1T Corpus (Brants and Franz, 2006), a large collection of n-grams and their counts created from approximately one trillion words of Web text. Only n-grams containing lowercase words were used. 5-grams were not used due to memory limitations. Only n-grams containing both terms (including plural forms) were extracted. Table 3 describes the extracted n-gram features.

5.2 Cross Validation Experiments

We performed 10-fold cross validation on our dataset, and, for the purpose of comparison, we also performed 5-fold cross validation on Ó Séaghdha's (2007) dataset using his folds. Our classification accuracy results are 79.3% on our data and 63.6% on the Ó Séaghdha data. We used the χ^2 measure to limit our experiments to the most useful 35000 features, which is the point where we obtain the highest results on Ó Séaghdha's data. The 63.6% figure is similar to the best previously reported accuracy for this dataset of 63.1%, which was obtained by Ó Séaghdha and Copestake (2009) using kernel methods.

For comparison with SVMs, we used Thorsten Joachims' $SVM^{multiclass}$, which implements an optimization solution to Cramer and Singer's (2001) multiclass SVM formulation. The best results were similar, with 79.4% on our dataset and 63.1% on Ó Séaghdha's. $SVM^{multiclass}$ was, however, observed to be very sensitive to the tuning of the C parameter, which determines the tradeoff between training error and margin width. The best results for the datasets were produced with C set to 5000 and 375 respectively.

Trigram Feature Extraction Patterns			
text	<n_1>	<n_2>	
<*>	<n_1>	<n_2>	
<n_1>	<n_2>	text	
<n_1>	<n_2>	<*>	
<n_1>	text	<n_2>	
<n_2>	text	<n_1>	
<n_1>	<*>	<n_2>	
<n_2>	<*>	<n_1>	
4-Gram Feature Extraction Patterns			
<n_1>	<n_2>	text	text
<n_1>	<n_2>	<*>	text
text	<n_1>	<n_2>	text
text	text	<n_1>	<n_2>
text	<*>	<n_1>	<n_2>
<n_1>	text	text	<n_2>
<n_1>	text	<*>	<n_2>
<n_1>	<*>	text	<n_2>
<n_1>	<*>	<*>	<n_2>
<n_2>	text	text	<n_1>
<n_2>	text	<*>	<n_1>
<n_2>	<*>	text	<n_1>
<n_2>	<*>	<*>	<n_1>

Table 3: Patterns for extracting trigram and 4-Gram features from the Web 1T Corpus for a given noun compound (n_1 n_2).

To assess the impact of the various features, we ran the cross validation experiments for each feature type, alternating between including *only* one

feature type and including all feature types *except* that one. The results for these runs using the Maximum Entropy classifier are presented in Table 4.

There are several points of interest in these results. The WordNet gloss terms had a surprisingly strong influence. In fact, by themselves they proved roughly as useful as the hypernym features, and their removal had the single strongest negative impact on accuracy for our dataset. As far as we know, this is the first time that WordNet definition words have been used as features for noun compound interpretation. In the future, it may be valuable to add definition words from other machine-readable dictionaries. The influence of the Web 1T n-gram features was somewhat mixed. They had a positive impact on the Ó Séaghdha data, but their affect upon our dataset was limited and mixed, with the removal of the 4-gram features actually improving performance slightly.

	Our Data		Ó Séaghdha Data	
	1	M-1	1	M-1
WordNet-based				
synonyms	0.674	0.793	0.469	0.626
hypernyms	0.753	0.787	0.539	0.626
hypernyms$_\cap$	0.250	0.791	0.357	0.624
gloss terms	0.741	0.785	0.510	0.613
gloss terms$_\cap$	0.226	0.793	0.275	0.632
lexfnames	0.583	0.792	0.505	0.629
lexfnames$_\wedge$	0.480	0.790	0.440	0.629
linktypes	0.328	0.793	0.365	0.631
linktypes$_\wedge$	0.277	0.792	0.346	0.626
pos	0.146	0.793	0.239	0.633
pos$_\wedge$	0.146	0.793	0.235	0.632
part-of terms	0.372	0.793	0.368	0.635
lexicalized	0.132	0.793	0.213	0.637
part of other	0.132	0.793	0.216	0.636
gloss of other	0.133	0.793	0.214	0.635
hypernym of other	0.132	0.793	0.227	0.627
Roget's Thesaurus-based				
div info	0.679	0.789	0.471	0.629
div info$_\cap$	0.173	0.793	0.283	0.633
Surface level				
affixes	0.200	0.793	0.274	0.637
affixes$_\wedge$	0.201	0.792	0.272	0.635
last letters	0.481	0.792	0.396	0.634
prepositions	0.136	0.793	0.222	0.635
Web 1T-based				
trigrams	0.571	0.790	0.437	0.615
4-grams	0.558	0.797	0.442	0.604

Table 4: Impact of features; cross validation accuracy for *only one feature type* and *all but one feature type* experiments, denoted by 1 and M-1 respectively. $\cap$–features shared by both n_1 and n_2; $\wedge$–n_1 and n_2 features conjoined by logical AND (e.g., n_1 is a 'substance' $\wedge$ n_2 is a 'artifact')

6 Evaluation

6.1 Evaluation Data

To assess the quality of our taxonomy and classification method, we performed an inter-annotator agreement study using 150 noun compounds extracted from a random subset of articles taken from New York Times articles dating back to 1987 (Sandhaus, 2008). The terms were selected based upon their frequency (i.e., a compound occurring twice as often as another is twice as likely to be selected) to label for testing purposes. Using a heuristic similar to that used by Lauer (1995), we only extracted binary noun compounds not part of a larger sequence. Before reaching the 150 mark, we discarded 94 of the drawn examples because they were included in the training set. Thus, our training set covers roughly 38.5% of the binary noun compound *instances* in recent New York Times articles.

6.2 Annotators

Due to the relatively high speed and low cost of Amazon's Mechanical Turk service, we chose to use Mechanical Turkers as our annotators.

Using Mechanical Turk to obtain inter-annotator agreement figures has several drawbacks. The first and most significant drawback is that it is impossible to force each Turker to label every data point without putting all the terms onto a single web page, which is highly impractical for a large taxonomy. Some Turkers may label every compound, but most do not. Second, while we requested that Turkers only work on our task if English was their first language, we had no method of enforcing this. Third, Turker annotation quality varies considerably.

6.3 Combining Annotators

To overcome the shortfalls of using Turkers for an inter-annotator agreement study, we chose to request ten annotations per noun compound and then combine the annotations into a single set of selections using a weighted voting scheme. To combine the results, we calculated a "quality" score for each Turker based upon how often he/she agreed with the others. This score was computed as the average percentage of other Turkers who agreed with his/her annotations. The score for each label for a particular compound was then computed as the sum of the Turker quality scores of the Turkers

who annotated the compound. Finally, the label with the highest rating was selected.

6.4 Inter-annotator Agreement Results

The raw agreement scores along with Cohen's κ (Cohen, 1960), a measure of inter-annotator agreement that discounts random chance, were calculated against the authors' labeling of the data for each Turker, the weighted-voting annotation set, and the automatic classification output. These statistics are reported in Table 5 along with the individual Turker "quality" scores. The 54 Turkers who made fewer than 3 annotations were excluded from the calculations under the assumption that they were not dedicated to the task, leaving a total of 49 Turkers. Due to space limitations, only results for Turkers who annotated 15 or more instances are included in Table 5.

We recomputed the κ statistics after conflating the category groups in two different ways. The first variation involved conflating all the TOPIC categories into a single topic category, resulting in a total of 37 categories (denoted by κ* in Table 5). For the second variation, in addition to conflating the TOPIC categories, we conflated the ATTRIBUTE categories into a single category and the PURPOSE/ACTIVITY categories into a single category, for a total of 27 categories (denoted by κ** in Table 5).

Id	N	Weight	Agree	κ	κ*	κ**
1	23	0.45	0.70	0.67	0.67	0.74
2	34	0.46	0.68	0.65	0.65	0.72
3	35	0.34	0.63	0.60	0.61	0.61
4	24	0.46	0.63	0.59	0.68	0.76
5	16	0.58	0.63	0.59	0.59	0.54
Voted	150	NA	0.59	0.57	0.61	0.67
6	52	0.45	0.58	0.54	0.60	0.60
7	38	0.35	0.55	0.52	0.54	0.56
8	149	0.36	0.52	0.49	0.53	0.58
Auto	150	NA	0.51	0.47	0.47	0.45
9	88	0.38	0.48	0.45	0.49	0.59
10	36	0.42	0.47	0.43	0.48	0.52
11	104	0.29	0.46	0.43	0.48	0.52
12	38	0.33	0.45	0.40	0.46	0.47
13	66	0.31	0.42	0.39	0.39	0.49
14	15	0.27	0.40	0.34	0.31	0.29
15	62	0.23	0.34	0.29	0.35	0.38
16	150	0.23	0.30	0.26	0.26	0.30
17	19	0.24	0.26	0.21	0.17	0.14
18	144	0.21	0.25	0.20	0.22	0.22
19	29	0.18	0.21	0.14	0.17	0.31
20	22	0.18	0.18	0.12	0.10	0.16
21	51	0.19	0.18	0.13	0.20	0.26
22	41	0.02	0.02	0.00	0.00	0.01

Table 5: Annotation results. Id – annotator id; N – number of annotations; Weight – voting weight; Agree – raw agreement versus the author's annotations; κ – Cohen's κ agreement; κ* and κ** – Cohen's κ results after conflating certain categories. Voted – combined annotation set using weighted voting; Auto – automatic classification output.

6.5 Results Discussion

The .57-.67 κ figures achieved by the Voted annotations compare well with previously reported inter-annotator agreement figures for noun compounds using fine-grained taxonomies. Kim and Baldwin (2005) report an agreement of 52.31% (not κ) for their dataset using Barker and Szpakowicz's (1998) 20 semantic relations. Girju et al. (2005) report .58 κ using a set of 35 semantic relations, only 21 of which were used, and a .80 κ score using Lauer's 8 prepositional paraphrases. Girju (2007) reports .61 κ agreement using a similar set of 22 semantic relations for noun compound annotation in which the annotators are shown translations of the compound in foreign languages. Ó Séaghdha (2007) reports a .68 κ for a relatively small set of relations (BE, HAVE, IN, INST, ACTOR, ABOUT) after removing compounds with non-specific associations or high lexicalization. The correlation between our automatic "quality" scores for the Turkers who performed at least three annotations and their simple agreement with our annotations was very strong at 0.88.

The .51 automatic classification figure is respectable given the larger number of categories in the taxonomy. It is also important to remember that the training set covers a large portion of the two-word noun compound instances in recent New York Times articles, so substantially higher accuracy can be expected on many texts. Interestingly, conflating categories only improved the κ statistics for the Turkers, not the automatic classifier.

7 Conclusion

In this paper, we present a novel, fine-grained taxonomy of 43 noun-noun semantic relations, the largest annotated noun compound dataset yet created, and a supervised classification method for automatic noun compound interpretation.

We describe our taxonomy and provide mappings to taxonomies used by others. Our inter-annotator agreement study, which utilized non-experts, shows good inter-annotator agreement

given the difficulty of the task, indicating that our category definitions are relatively straightforward. Our taxonomy provides wide coverage, with only 2.32% of our dataset marked as other/lexicalized and 2.67% of our 150 inter-annotator agreement data marked as such by the combined Turker (Voted) annotation set.

We demonstrated the effectiveness of a straightforward, supervised classification approach to noun compound interpretation that uses a large variety of boolean features. We also examined the importance of the different features, noting a novel and very useful set of features—the words comprising the definitions of the individual words.

8 Future Work

In the future, we plan to focus on the interpretation of noun compounds with 3 or more nouns, a problem that includes bracketing noun compounds into their dependency structures in addition to noun-noun semantic relation interpretation. Furthermore, we would like to build a system that can handle longer noun phrases, including prepositions and possessives.

We would like to experiment with including features from various other lexical resources to determine their usefulness for this problem.

Eventually, we would like to expand our data set and relations to cover proper nouns as well. We are hopeful that our current dataset and relation definitions, which will be made available via http://www.isi.edu will be helpful to other researchers doing work regarding text semantics.

Acknowledgements

Stephen Tratz is supported by a National Defense Science and Engineering Graduate Fellowship.

References

Ahn, K., J. Bos, J. R. Curran, D. Kor, M. Nissim, and B. Webber. 2005. Question Answering with QED at TREC-2005. In *Proc. of TREC-2005*.

Baldwin, T. & T. Tanaka 2004. Translation by machine of compound nominals: Getting it right. In *Proc. of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing*.

Barker, K. and S. Szpakowicz. 1998. Semi-Automatic Recognition of Noun Modifier Relationships. In *Proc. of the 17th International Conference on Computational Linguistics*.

Berger, A., S. A. Della Pietra, and V. J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics* 22:39-71.

Brants, T. and A. Franz. 2006. Web 1T 5-gram Corpus Version 1.1. Linguistic Data Consortium.

Butnariu, C. and T. Veale. 2008. A concept-centered approach to noun-compound interpretation. In *Proc. of 22nd International Conference on Computational Linguistics (COLING 2008)*.

Butnariu, C., S.N. Kim, P. Nakov, D. Ó Séaghdha, S. Szpakowicz, and T. Veale. 2009. SemEval Task 9: The Interpretation of Noun Compounds Using Paraphrasing Verbs and Prepositions. In *Proc. of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions*.

Cohen, J. 1960. A coefficient of agreement for nominal scales. Educational and Psychological Measurement. 20:1.

Crammer, K. and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs In *Journal of Machine Learning Research*.

Downing, P. 1977. On the Creation and Use of English Compound Nouns. Language. 53:4.

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

Finin, T. 1980. The Semantic Interpretation of Compound Nominals. *Ph.D dissertation* University of Illinois, Urbana, Illinois.

Girju, R., D. Moldovan, M. Tatu and D. Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19.

Girju, R. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *Proc. of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*.

Girju, R. 2009. The Syntax and Semantics of Prepositions in the Task of Automatic Interpretation of Nominal Phrases and Compounds: a Cross-linguistic Study. In *Computational Linguistics 35(2) - Special Issue on Prepositions in Application*.

Jespersen, O. 1949. A Modern English Grammar on Historical Principles. Ejnar Munksgaard. Copenhagen.

Kim, S.N. and T. Baldwin. 2007. Interpreting Noun Compounds using Bootstrapping and Sense Collocation. In *Proc. of the 10th Conf. of the Pacific Association for Computational Linguistics*.

Kim, S.N. and T. Baldwin. 2005. Automatic Interpretation of Compound Nouns using WordNet::Similarity. In *Proc. of 2nd International Joint Conf. on Natural Language Processing*.

Lauer, M. 1995. Corpus statistics meet the compound noun. In *Proc. of the 33rd Meeting of the Association for Computational Linguistics*.

Lees, R.B. 1960. The Grammar of English Nominalizations. Indiana University. Bloomington, IN.

Levi, J.N. 1978. The Syntax and Semantics of Complex Nominals. Academic Press. New York.

McCallum, A. K. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu. 2002.

Moldovan, D., A. Badulescu, M. Tatu, D. Antohe, and R. Girju. 2004. Models for the semantic classification of noun phrases. In *Proc. of Computational Lexical Semantics Workshop at HLT-NAACL 2004.*

Nakov, P. and M. Hearst. 2005. Search Engine Statistics Beyond the n-gram: Application to Noun Compound Bracketing. In *Proc. the Ninth Conference on Computational Natural Language Learning*.

Nakov, P. 2008. Noun Compound Interpretation Using Paraphrasing Verbs: Feasibility Study. In *Proc. the 13th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'08)*.

Nastase V. and S. Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proc. the 5th International Workshop on Computational Semantics*.

Nastase, V., J. S. Shirabad, M. Sokolova, and S. Szpakowicz 2006. Learning noun-modifier semantic relations with corpus-based and Wordnet-based features. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06)*.

Ó Séaghdha, D. and A. Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*.

Ó Séaghdha, D. 2007. Annotating and Learning Compound Noun Semantics. In *Proc. of the ACL 2007 Student Research Workshop*.

Rosario, B. and M. Hearst. 2001. Classifying the Semantic Relations in Noun Compounds via Domain-Specific Lexical Hierarchy. In *Proc. of 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*.

Sandhaus, E. 2008. The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia.

Spärck Jones, K. 1983. Compound Noun Interpretation Problems. *Computer Speech Processing*, eds. F. Fallside and W A. Woods, Prentice-Hall, NJ.

Turney, P. D. 2006. Similarity of semantic relations. *Computation Linguistics*, 32(3):379-416

Vanderwende, L. 1994. Algorithm for Automatic Interpretation of Noun Sequences. In *Proc. of COLING-94*.

Warren, B. 1978. Semantic Patterns of Noun-Noun Compounds. Acta Universitatis Gothobugensis.

Ye, P. and T. Baldwin. 2007. MELB-YB: Preposition Sense Disambiguation Using Rich Semantic Features. In *Proc. of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*.

Models of Metaphor in NLP

Ekaterina Shutova
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
Ekaterina.Shutova@cl.cam.ac.uk

Abstract

Automatic processing of metaphor can be clearly divided into two subtasks: *metaphor recognition* (distinguishing between literal and metaphorical language in a text) and *metaphor interpretation* (identifying the intended literal meaning of a metaphorical expression). Both of them have been repeatedly addressed in NLP. This paper is the first comprehensive and systematic review of the existing computational models of metaphor, the issues of metaphor annotation in corpora and the available resources.

1 Introduction

Our production and comprehension of language is a multi-layered computational process. Humans carry out high-level semantic tasks effortlessly by subconsciously employing a vast inventory of complex linguistic devices, while simultaneously integrating their background knowledge, to reason about reality. An ideal model of language understanding would also be capable of performing such high-level semantic tasks.

However, a great deal of NLP research to date focuses on processing lower-level linguistic information, such as e.g. part-of-speech tagging, discovering syntactic structure of a sentence (parsing), coreference resolution, named entity recognition and many others. Another cohort of researchers set the goal of improving application-based statistical inference (e.g. for recognizing textual entailment or automatic summarization). In contrast, there have been fewer attempts to bring the state-of-the-art NLP technologies together to model the way humans use language to frame high-level reasoning processes, such as for example, creative thought.

The majority of computational approaches to figurative language still exploit the ideas articulated three decades ago (Wilks, 1978; Lakoff and Johnson, 1980; Fass, 1991) and often rely on task-specific hand-coded knowledge. However, recent work on lexical semantics and lexical acquisition techniques opens many new avenues for creation of fully automated models for recognition and interpretation of figurative language. In this paper I will focus on the phenomenon of metaphor and describe the most prominent computational approaches to metaphor, as well the issues of resource creation and metaphor annotation.

Metaphors arise when one concept is viewed in terms of the properties of the other. In other words it is based on *similarity* between the concepts. Similarity is a kind of association implying the presence of characteristics in common. Here are some examples of metaphor.

(1) Hillary *brushed aside* the accusations.

(2) How can I *kill* a process? (Martin, 1988)

(3) I *invested* myself fully in this relationship.

(4) And then my heart with pleasure *fills*,
And *dances* with the daffodils.[1]

In metaphorical expressions seemingly unrelated features of one concept are associated with another concept. In the example (2) the *computational process* is viewed as something *alive* and, therefore, its forced termination is associated with the act of killing.

Metaphorical expressions represent a great variety, ranging from conventional metaphors, which we reproduce and comprehend every day, e.g. those in (2) and (3), to poetic and largely novel ones, such as (4). The use of metaphor is ubiquitous in natural language text and it is a serious bottleneck in automatic text understanding.

[1] "I wandered lonely as a cloud", William Wordsworth, 1804.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 688–697,
Uppsala, Sweden, 11-16 July 2010.

In order to estimate the frequency of the phenomenon, Shutova (2010) conducted a corpus study on a subset of the British National Corpus (BNC) (Burnard, 2007) representing various genres. They manually annotated metaphorical expressions in this data and found that 241 out of 761 sentences contained a metaphor. Due to such a high frequency of their use, a system capable of recognizing and interpreting metaphorical expressions in unrestricted text would become an invaluable component of any semantics-oriented NLP application.

Automatic processing of metaphor can be clearly divided into two subtasks: *metaphor recognition* (distinguishing between literal and metaphorical language in text) and *metaphor interpretation* (identifying the intended literal meaning of a metaphorical expression). Both of them have been repeatedly addressed in NLP.

2 Theoretical Background

Four different views on metaphor have been broadly discussed in linguistics and philosophy: the comparison view (Gentner, 1983), the interaction view (Black, 1962), (Hesse, 1966), the selectional restrictions violation view (Wilks, 1975; Wilks, 1978) and the conceptual metaphor view (Lakoff and Johnson, 1980)[2]. All of these approaches share the idea of an interconceptual mapping that underlies the production of metaphorical expressions. In other words, metaphor always involves two concepts or conceptual domains: the *target* (also called *topic* or *tenor* in the linguistics literature) and the *source* (or *vehicle*). Consider the examples in (5) and (6).

(5) He *shot down* all of my arguments. (Lakoff and Johnson, 1980)

(6) He *attacked* every weak point in my argument. (Lakoff and Johnson, 1980)

According to Lakoff and Johnson (1980), a mapping of a concept of *argument* to that of *war* is employed here. The *argument*, which is the target concept, is viewed in terms of a *battle* (or a *war*), the source concept. The existence of such a link allows us to talk about arguments using the war terminology, thus giving rise to a number of metaphors.

[2] A detailed overview and criticism of these four views can be found in (Tourangeau and Sternberg, 1982).

However, Lakoff and Johnson do not discuss how metaphors can be recognized in the linguistic data, which is the primary task in the automatic processing of metaphor. Although humans are highly capable of producing and comprehending metaphorical expressions, the task of distinguishing between literal and non-literal meanings and, therefore, identifying metaphor in text appears to be challenging. This is due to the variation in its use and external form, as well as a not clear-cut semantic distinction. Gibbs (1984) suggests that literal and figurative meanings are situated at the ends of a single continuum, along which metaphoricity and idiomaticity are spread. This makes demarcation of metaphorical and literal language fuzzy.

So far, the most influential account of metaphor recognition is that of Wilks (1978). According to Wilks, metaphors represent a violation of *selectional restrictions* in a given context. Selectional restrictions are the semantic constraints that a verb places onto its arguments. Consider the following example.

(7) My car *drinks* gasoline. (Wilks, 1978)

The verb *drink* normally takes an *animate* subject and a *liquid* object. Therefore, *drink* taking a *car* as a subject is an anomaly, which may in turn indicate the metaphorical use of *drink*.

3 Automatic Metaphor Recognition

One of the first attempts to identify and interpret metaphorical expressions in text automatically is the approach of Fass (1991). It originates in the work of Wilks (1978) and utilizes hand-coded knowledge. Fass (1991) developed a system called met*, capable of discriminating between literalness, metonymy, metaphor and anomaly. It does this in three stages. First, literalness is distinguished from non-literalness using selectional preference violation as an indicator. In the case that non-literalness is detected, the respective phrase is tested for being a metonymic relation using hand-coded patterns (such as CONTAINER-for-CONTENT). If the system fails to recognize metonymy, it proceeds to search the knowledge base for a *relevant analogy* in order to discriminate metaphorical relations from anomalous ones. E.g., the sentence in (7) would be represented in this framework as (*car,drink,gasoline*), which does not satisfy the preference (*animal,drink,liquid*), as *car*

is not a hyponym of *animal*. met* then searches its knowledge base for a triple containing a hypernym of both the actual argument and the desired argument and finds (*thing,use,energy_source*), which represents the metaphorical interpretation.

However, Fass himself indicated a problem with the selectional preference violation approach applied to metaphor recognition. The approach detects any kind of non-literalness or anomaly in language (metaphors, metonymies and others), and not only metaphors, i.e., it overgenerates. The methods met* uses to differentiate between those are mainly based on hand-coded knowledge, which implies a number of limitations.

Another problem with this approach arises from the high conventionality of metaphor in language. This means that some metaphorical senses are very common. As a result the system would extract selectional preference distributions skewed towards such conventional metaphorical senses of the verb or one of its arguments. Therefore, although some expressions may be fully metaphorical in nature, no selectional preference violation can be detected in their use. Another counterargument is bound to the fact that interpretation is always context dependent, e.g. the phrase *all men are animals* can be used metaphorically, however, without any violation of selectional restrictions.

Goatly (1997) addresses the phenomenon of metaphor by identifying a set of linguistic cues indicating it. He gives examples of lexical patterns indicating the presence of a metaphorical expression, such as *metaphorically speaking, utterly, completely, so to speak* and, surprisingly, *literally*. Such cues would probably not be enough for metaphor extraction on their own, but could contribute to a more complex system.

The work of Peters and Peters (2000) concentrates on detecting figurative language in lexical resources. They mine WordNet (Fellbaum, 1998) for the examples of systematic polysemy, which allows to capture metonymic and metaphorical relations. The authors search for nodes that are relatively high up in the WordNet hierarchy and that share a set of common word forms among their descendants. Peters and Peters found that such nodes often happen to be in metonymic (e.g. *publication – publisher*) or metaphorical (e.g. *supporting structure – theory*) relation.

The CorMet system discussed in (Mason, 2004) is the first attempt to discover source-target domain mappings automatically. This is done by "finding systematic variations in domain-specific selectional preferences, which are inferred from large, dynamically mined Internet corpora". For example, Mason collects texts from the LAB domain and the FINANCE domain, in both of which *pour* would be a characteristic verb. In the LAB domain *pour* has a strong selectional preference for objects of type *liquid*, whereas in the FINANCE domain it selects for *money*. From this Mason's system infers the domain mapping FINANCE – LAB and the concept mapping *money – liquid*. He compares the output of his system against the Master Metaphor List (Lakoff et al., 1991) containing hand-crafted metaphorical mappings between concepts. Mason reports an accuracy of 77%, although it should be noted that as any evaluation that is done by hand it contains an element of subjectivity.

Birke and Sarkar (2006) present a sentence clustering approach for non-literal language recognition implemented in the TroFi system (Trope Finder). This idea originates from a similarity-based word sense disambiguation method developed by Karov and Edelman (1998). The method employs a set of seed sentences, where the senses are annotated; computes similarity between the sentence containing the word to be disambiguated and all of the seed sentences and selects the sense corresponding to the annotation in the most similar seed sentences. Birke and Sarkar (2006) adapt this algorithm to perform a two-way classification: literal vs. non-literal, and they do not clearly define the kinds of tropes they aim to discover. They attain a performance of 53.8% in terms of f-score.

The method of Gedigan et al. (2006) discriminates between literal and metaphorical use. They trained a maximum entropy classifier for this purpose. They obtained their data by extracting the lexical items whose frames are related to MOTION and CURE from FrameNet (Fillmore et al., 2003). Then they searched the PropBank Wall Street Journal corpus (Kingsbury and Palmer, 2002) for sentences containing such lexical items and annotated them with respect to metaphoricity. They used PropBank annotation (arguments and their semantic types) as features to train the classifier and report an accuracy of 95.12%. This result is, however, only a little higher than the performance of the naive baseline assigning majority class to all instances (92.90%). These numbers

can be explained by the fact that 92.00% of the verbs of MOTION and CURE in the Wall Street Journal corpus are used metaphorically, thus making the dataset unbalanced with respect to the target categories and the task notably easier.

Both Birke and Sarkar (2006) and Gedigan et al. (2006) focus only on metaphors expressed by a verb. As opposed to that the approach of Krishnakumaran and Zhu (2007) deals with verbs, nouns and adjectives as parts of speech. They use hyponymy relation in WordNet and word bigram counts to predict metaphors at a sentence level. Given an IS-A metaphor (e.g. *The world is a stage*[3]) they verify if the two nouns involved are in hyponymy relation in WordNet, and if they are not then this sentence is tagged as containing a metaphor. Along with this they consider expressions containing a verb or an adjective used metaphorically (e.g. *He planted good ideas in their minds* or *He has a fertile imagination*). Hereby they calculate bigram probabilities of verb-noun and adjective-noun pairs (including the hyponyms/hypernyms of the noun in question). If the combination is not observed in the data with sufficient frequency, the system tags the sentence containing it as metaphorical. This idea is a modification of the selectional preference view of Wilks. However, by using bigram counts over verb-noun pairs Krishnakumaran and Zhu (2007) loose a great deal of information compared to a system extracting verb-object relations from parsed text. The authors evaluated their system on a set of example sentences compiled from the Master Metaphor List (Lakoff et al., 1991), whereby highly conventionalized metaphors (they call them *dead metaphors*) are taken to be negative examples. Thus they do not deal with literal examples as such: essentially, the distinction they are making is between the senses included in WordNet, even if they are conventional metaphors, and those not included in WordNet.

4 Automatic Metaphor Interpretation

Almost simultaneously with the work of Fass (1991), Martin (1990) presents a Metaphor Interpretation, Denotation and Acquisition System (MIDAS). In this work Martin captures hierarchical organisation of conventional metaphors. The idea behind this is that the more specific conventional metaphors descend from the general ones. Given an example of a metaphorical expression, MIDAS searches its database for a corresponding metaphor that would explain the anomaly. If it does not find any, it abstracts from the example to more general concepts and repeats the search. If it finds a suitable general metaphor, it creates a mapping for its descendant, a more specific metaphor, based on this example. This is also how novel metaphors are acquired. MIDAS has been integrated with the Unix Consultant (UC), the system that answers users questions about Unix. The UC first tries to find a literal answer to the question. If it is not able to, it calls MIDAS which detects metaphorical expressions via selectional preference violation and searches its database for a metaphor explaining the anomaly in the question.

Another cohort of approaches relies on performing inferences about entities and events in the source and target domains for metaphor interpretation. These include the KARMA system (Narayanan, 1997; Narayanan, 1999; Feldman and Narayanan, 2004) and the ATT-Meta project (Barnden and Lee, 2002; Agerri et al., 2007). Within both systems the authors developed a metaphor-based reasoning framework in accordance with the theory of conceptual metaphor. The reasoning process relies on manually coded knowledge about the world and operates mainly in the source domain. The results are then projected onto the target domain using the conceptual mapping representation. The ATT-Meta project concerns metaphorical and metonymic description of mental states and reasoning about mental states using first order logic. Their system, however, does not take natural language sentences as input, but logical expressions that are representations of small discourse fragments. KARMA in turn deals with a broad range of abstract actions and events and takes parsed text as input.

Veale and Hao (2008) derive a "fluid knowledge representation for metaphor interpretation and generation", called Talking Points. Talking Points are a set of characteristics of concepts belonging to source and target domains and related facts about the world which the authors acquire automatically from WordNet and from the web. Talking Points are then organized in *Slipnet*, a framework that allows for a number of insertions, deletions and substitutions in definitions of such characteristics in order to establish a connection between the target and the source

[3] William Shakespeare

concepts. This work builds on the idea of *slippage* in knowledge representation for understanding analogies in abstract domains (Hofstadter and Mitchell, 1994; Hofstadter, 1995). Below is an example demonstrating how slippage operates to explain the metaphor *Make-up is a Western burqa*.

Make-up =>
≡ typically worn by women
≈ expected to be worn by women
≈ must be worn by women
≈ must be worn by Muslim women
Burqa <=

By doing insertions and substitutions the system arrives from the definition *typically worn by women* to that of *must be worn by Muslim women*, and thus establishes a link between the concepts of *make-up* and *burqa*. Veale and Hao (2008), however, did not evaluate to which extent their knowledge base of Talking Points and the associated reasoning framework are useful to interpret metaphorical expressions occurring in text.

Shutova (2010) defines metaphor interpretation as a paraphrasing task and presents a method for deriving literal paraphrases for metaphorical expressions from the BNC. For example, for the metaphors in "All of this *stirred* an unfathomable excitement in her" or "a carelessly *leaked* report" their system produces interpretations "All of this provoked an unfathomable excitement in her" and "a carelessly disclosed report" respectively. They first apply a probabilistic model to rank all possible paraphrases for the metaphorical expression given the context; and then use automatically induced selectional preferences to discriminate between figurative and literal paraphrases. The selectional preference distribution is defined in terms of selectional association measure introduced by Resnik (1993) over the noun classes automatically produced by Sun and Korhonen (2009). Shutova (2010) tested their system only on metaphors expressed by a verb and report a paraphrasing accuracy of 0.81.

5 Metaphor Resources

Metaphor is a knowledge-hungry phenomenon. Hence there is a need for either an extensive manually-created knowledge-base or a robust knowledge acquisition system for interpretation of metaphorical expressions. The latter being a hard task, a great deal of metaphor research resorted to the first option. Although hand-coded knowledge proved useful for metaphor interpretation (Fass, 1991; Martin, 1990), it should be noted that the systems utilizing it have a very limited coverage.

One of the first attempts to create a multi-purpose knowledge base of source–target domain mappings is the Master Metaphor List (Lakoff et al., 1991). It includes a classification of metaphorical mappings (mainly those related to mind, feelings and emotions) with the corresponding examples of language use. This resource has been criticized for the lack of clear structuring principles of the mapping ontology (Lönneker-Rodman, 2008). The taxonomical levels are often confused, and the same classes are referred to by different class labels. This fact and the chosen data representation in the Master Metaphor List make it not suitable for computational use. However, both the idea of the list and its actual mappings ontology inspired the creation of other metaphor resources.

The most prominent of them are MetaBank (Martin, 1994) and the Mental Metaphor Databank[4] created in the framework of the ATT-meta project (Barnden and Lee, 2002; Agerri et al., 2007). The MetaBank is a knowledge-base of English metaphorical conventions, represented in the form of *metaphor maps* (Martin, 1988) containing detailed information about source-target concept mappings backed by empirical evidence. The ATT-meta project databank contains a large number of examples of metaphors of mind classified by source–target domain mappings taken from the Master Metaphor List.

Along with this it is worth mentioning metaphor resources in languages other than English. There has been a wealth of research on metaphor in Spanish, Chinese, Russian, German, French and Italian. The Hamburg Metaphor Database (Lönneker, 2004; Reining and Lönneker-Rodman, 2007) contains examples of metaphorical expressions in German and French, which are mapped to senses from EuroWordNet[5] and annotated with source–target domain mappings taken from the Master Metaphor List.

Alonge and Castelli (2003) discuss how metaphors can be represented in ItalWordNet for

[4] http://www.cs.bham.ac.uk/∼jab/ATT-Meta/Databank/

[5] EuroWordNet is a multilingual database with wordnets for several European languages (Dutch, Italian, Spanish, German, French, Czech and Estonian). The wordnets are structured in the same way as the Princeton WordNet for English. URL: http://www.illc.uva.nl/EuroWordNet/

Italian and motivate this by linguistic evidence. Encoding metaphorical information in general-domain lexical resources for English, e.g. WordNet (Lönneker and Eilts, 2004), would undoubtedly provide a new platform for experiments and enable researchers to directly compare their results.

6 Metaphor Annotation in Corpora

To reflect two distinct aspects of the phenomenon, metaphor annotation can be split into two stages: identifying metaphorical senses in text (akin word sense disambiguation) and annotating source – target domain mappings underlying the production of metaphorical expressions. Traditional approaches to metaphor annotation include manual search for lexical items used metaphorically (Pragglejaz Group, 2007), for source and target domain vocabulary (Deignan, 2006; Koivisto-Alanko and Tissari, 2006; Martin, 2006) or for linguistic markers of metaphor (Goatly, 1997). Although there is a consensus in the research community that the phenomenon of metaphor is not restricted to similarity-based extensions of meanings of isolated words, but rather involves reconceptualization of a whole area of experience in terms of another, there still has been surprisingly little interest in annotation of cross-domain mappings. However, a corpus annotated for conceptual mappings could provide a new starting point for both linguistic and cognitive experiments.

6.1 Metaphor and Polysemy

The theorists of metaphor distinguish between two kinds of metaphorical language: *novel* (or *poetic*) metaphors, that surprise our imagination, and *conventionalized* metaphors, that become a part of an ordinary discourse. "Metaphors begin their lives as novel poetic creations with marked rhetorical effects, whose comprehension requires a special imaginative leap. As time goes by, they become a part of general usage, their comprehension becomes more automatic, and their rhetorical effect is dulled" (Nunberg, 1987). Following Orwell (1946) Nunberg calls such metaphors "dead" and claims that they are not psychologically distinct from literally-used terms.

This scheme demonstrates how metaphorical associations capture some generalisations governing polysemy: over time some of the aspects of the target domain are added to the meaning of a term in a source domain, resulting in a (metaphorical) sense extension of this term. Copestake and Briscoe (1995) discuss sense extension mainly based on metonymic examples and model the phenomenon using lexical rules encoding metonymic patterns. Along with this they suggest that similar mechanisms can be used to account for metaphoric processes, and the conceptual mappings encoded in the sense extension rules would define the limits to the possible shifts in meaning.

However, it is often unclear if a metaphorical instance is a case of broadening of the sense in context due to general vagueness in language, or it manifests a formation of a new distinct metaphorical sense. Consider the following examples.

(8) a. As soon as I *entered* the room I noticed the difference.

b. How can I *enter* Emacs?

(9) a. My tea is *cold*.

b. He is such a *cold* person.

Enter in (8a) is defined as "to go or come into a place, building, room, etc.; to pass within the boundaries of a country, region, portion of space, medium, etc."[6] In (8b) this sense stretches to describe dealing with *software*, whereby COMPUTER PROGRAMS are viewed as PHYSICAL SPACES. However, this extended sense of *enter* does not appear to be sufficiently distinct or conventional to be included into the dictionary, although this could happen over time.

The sentence (9a) exemplifies the basic sense of *cold* – "of a temperature sensibly lower than that of the living human body", whereas *cold* in (9b) should be interpreted metaphorically as "void of ardour, warmth, or intensity of feeling; lacking enthusiasm, heartiness, or zeal; indifferent, apathetic". These two senses are clearly linked via the metaphoric mapping between EMOTIONAL STATES and TEMPERATURES.

A number of metaphorical senses are included in WordNet, however without any accompanying semantic annotation.

6.2 Metaphor Identification

6.2.1 Pragglejaz Procedure

Pragglejaz Group (2007) proposes a metaphor identification procedure (MIP) within the frame-

[6] Sense definitions are taken from the Oxford English Dictionary.

work of the Metaphor in Discourse project (Steen, 2007). The procedure involves metaphor annotation at the word level as opposed to identifying metaphorical relations (between words) or source–target domain mappings (between concepts or domains). In order to discriminate between the verbs used metaphorically and literally the annotators are asked to follow the guidelines:

1. For each verb establish its meaning in context and try to imagine a more basic meaning of this verb on other contexts. Basic meanings normally are: (1) more concrete; (2) related to bodily action; (3) more precise (as opposed to vague); (4) historically older.

2. If you can establish the basic meaning that is distinct from the meaning of the verb in this context, the verb is likely to be used metaphorically.

Such annotation can be viewed as a form of word sense disambiguation with an emphasis on metaphoricity.

6.2.2 Source – Target Domain Vocabulary

Another popular method that has been used to extract metaphors is searching for sentences containing lexical items from the source domain, the target domain, or both (Stefanowitsch, 2006). This method requires exhaustive lists of source and target domain vocabulary.

Martin (2006) conducted a corpus study in order to confirm that metaphorical expressions occur in text in contexts containing such lexical items. He performed his analysis on the data from the Wall Street Journal (WSJ) corpus and focused on four conceptual metaphors that occur with considerable regularity in the corpus. These include NUMERICAL VALUE AS LOCATION, COMMERCIAL ACTIVITY AS CONTAINER, COMMERCIAL ACTIVITY AS PATH FOLLOWING and COMMERCIAL ACTIVITY AS WAR. Martin manually compiled the lists of terms characteristic for each domain by examining sampled metaphors of these types and then augmented them through the use of thesaurus. He then searched the WSJ for sentences containing vocabulary from these lists and checked whether they contain metaphors of the above types. The goal of this study was to evaluate predictive ability of contexts containing vocabulary from (1) source domain and (2) target domain, as well as (3) estimating the likelihood of a metaphorical expression following another metaphorical expression described by the same mapping. He obtained the most positive results for metaphors of the type NUMERICAL-VALUE-AS-LOCATION ($P(Metaphor|Source) = 0.069$, $P(Metaphor|Target) = 0.677$, $P(Metaphor|Metaphor) = 0.703$).

6.3 Annotating Source and Target Domains

Wallington et al. (2003) carried out a metaphor annotation experiment in the framework of the ATT-Meta project. They employed two teams of annotators. Team A was asked to annotate "interesting stretches", whereby a phrase was considered interesting if (1) its significance in the document was non-physical, (2) it could have a physical significance in another context with a similar syntactic frame, (3) this physical significance was related to the abstract one. Team B had to annotate phrases according to their own intuitive definition of metaphor. Besides metaphorical expressions Wallington et al. (2003) attempted to annotate the involved source – target domain mappings. The annotators were given a set of mappings from the Master Metaphor List and were asked to assign the most suitable ones to the examples. However, the authors do not report the level of interannotator agreement nor the coverage of the mappings in the Master Metaphor List on their data.

Shutova and Teufel (2010) adopt a different approach to the annotation of source – target domain mappings. They do not rely on predefined mappings, but instead derive independent sets of most common source and target categories. They propose a two stage procedure, whereby the metaphorical expressions are first identified using MIP, and then the source domain (where the basic sense comes from) and the target domain (the given context) are selected from the lists of categories. Shutova and Teufel (2010) report interannotator agreement of 0.61 (κ).

7 Conclusion and Future Directions

The eighties and nineties provided us with a wealth of ideas on the structure and mechanisms of the phenomenon of metaphor. The approaches formulated back then are still highly influential, although their use of hand-coded knowledge is becoming increasingly less convincing. The last decade witnessed a high technological leap in

natural language computation, whereby manually crafted rules gradually give way to more robust corpus-based statistical methods. This is also the case for metaphor research. The latest developments in the lexical acquisition technology will in the near future enable fully automated corpus-based processing of metaphor.

However, there is still a clear need in a unified metaphor annotation procedure and creation of a large publicly available metaphor corpus. Given such a resource the computational work on metaphor is likely to proceed along the following lines: (1) automatic acquisition of an extensive set of valid metaphorical associations from linguistic data via statistical pattern matching; (2) using the knowledge of these associations for metaphor recognition in the unseen unrestricted text and, finally, (3) interpretation of the identified metaphorical expressions by deriving the closest literal paraphrase (a representation that can be directly embedded in other NLP applications to enhance their performance).

Besides making our thoughts more vivid and filling our communication with richer imagery, metaphors also play an important structural role in our cognition. Thus, one of the long term goals of metaphor research in NLP and AI would be to build a computational intelligence model accounting for the way metaphors organize our conceptual system, in terms of which we think and act.

Acknowledgments

I would like to thank Anna Korhonen and my reviewers for their most helpful feedback on this paper. The support of Cambridge Overseas Trust, who fully funds my studies, is gratefully acknowledged.

References

R. Agerri, J.A. Barnden, M.G. Lee, and A.M. Wallington. 2007. Metaphor, inference and domain-independent mappings. In *Proceedings of RANLP-2007*, pages 17–23, Borovets, Bulgaria.

A. Alonge and M. Castelli. 2003. Encoding information on metaphoric expressions in WordNet-like resources. In *Proceedings of the ACL 2003 Workshop on Lexicon and Figurative Language*, pages 10–17.

J.A. Barnden and M.G. Lee. 2002. An artificial intelligence approach to metaphor understanding. *Theoria et Historia Scientiarum*, 6(1):399–412.

J. Birke and A. Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *In Proceedings of EACL-06*, pages 329–336.

M. Black. 1962. *Models and Metaphors*. Cornell University Press.

L. Burnard. 2007. *Reference Guide for the British National Corpus (XML Edition)*.

A. Copestake and T. Briscoe. 1995. Semi-productive polysemy and sense extension. *Journal of Semantics*, 12:15–67.

A. Deignan. 2006. The grammar of linguistic metaphors. In A. Stefanowitsch and S. T. Gries, editors, *Corpus-Based Approaches to Metaphor and Metonymy*, Berlin. Mouton de Gruyter.

D. Fass. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

J. Feldman and S. Narayanan. 2004. Embodied meaning in a neural theory of language. *Brain and Language*, 89(2):385–392.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press, first edition.

C. J. Fillmore, C. R. Johnson, and M. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

M. Gedigan, J. Bryant, S. Narayanan, and B. Ciric. 2006. Catching metaphors. In *In Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, pages 41–48, New York.

D. Gentner. 1983. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170.

R. Gibbs. 1984. Literal meaning and psychological theory. *Cognitive Science*, 8:275–304.

A. Goatly. 1997. *The Language of Metaphors*. Routledge, London.

M. Hesse. 1966. *Models and Analogies in Science*. Notre Dame University Press.

D. Hofstadter and M. Mitchell. 1994. The Copycat Project: A model of mental fluidity and analogy-making. In K.J. Holyoak and J. A. Barnden, editors, *Advances in Connectionist and Neural Computation Theory*, Ablex, New Jersey.

D. Hofstadter. 1995. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. HarperCollins Publishers.

Y. Karov and S. Edelman. 1998. Similarity-based word sense disambiguation. *Computational Linguistics*, 24(1):41–59.

P. Kingsbury and M. Palmer. 2002. From TreeBank to PropBank. In *Proceedings of LREC-2002*, Gran Canaria, Canary Islands, Spain.

P. Koivisto-Alanko and H. Tissari. 2006. Sense and sensibility: Rational thought versus emotion in metaphorical language. In A. Stefanowitsch and S. T. Gries, editors, *Corpus-Based Approaches to Metaphor and Metonymy*, Berlin. Mouton de Gruyter.

S. Krishnakumaran and X. Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 13–20, Rochester, NY.

G. Lakoff and M. Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

G. Lakoff, J. Espenson, and A. Schwartz. 1991. The master metaphor list. Technical report, University of California at Berkeley.

B. Lönneker and C. Eilts. 2004. A Current Resource and Future Perspectives for Enriching WordNets with Metaphor Information. In *Proceedings of the Second International WordNet Conference—GWC 2004*, pages 157–162, Brno, Czech Republic.

B. Lönneker-Rodman. 2008. The hamburg metaphor database project: issues in resource creation. *Language Resources and Evaluation*, 42(3):293–318.

B. Lönneker. 2004. Lexical databases as resources for linguistic creativity: Focus on metaphor. In *Proceedings of the LREC 2004 Workshop on Language Resources for Linguistic Creativity*, pages 9–16, Lisbon, Portugal.

J. H. Martin. 1988. Representing regularities in the metaphoric lexicon. In *Proceedings of the 12th conference on Computational linguistics*, pages 396–401.

J. H. Martin. 1990. *A Computational Model of Metaphor Interpretation*. Academic Press Professional, Inc., San Diego, CA, USA.

J. H. Martin. 1994. Metabank: A knowledge-base of metaphoric language conventions. *Computational Intelligence*, 10:134–149.

J. H. Martin. 2006. A corpus-based analysis of context effects on metaphor comprehension. In A. Stefanowitsch and S. T. Gries, editors, *Corpus-Based Approaches to Metaphor and Metonymy*, Berlin. Mouton de Gruyter.

Z. J. Mason. 2004. Cormet: a computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44.

S. Narayanan. 1997. Knowledge-based action representations for metaphor and aspect (karma. Technical report, PhD thesis, University of California at Berkeley.

S. Narayanan. 1999. Moving right along: A computational model of metaphoric reasoning about events. In *Proceedings of AAAI 99)*, pages 121–128, Orlando, Florida.

G. Nunberg. 1987. Poetic and prosaic metaphors. In *Proceedings of the 1987 workshop on Theoretical issues in natural language processing*, pages 198–201.

G. Orwell. 1946. Politics and the english language. *Horizon*.

W. Peters and I. Peters. 2000. Lexicalised systematic polysemy in wordnet. In *Proceedings of LREC 2000*, Athens.

Pragglejaz Group. 2007. MIP: A method for identifying metaphorically used words in discourse. *Metaphor and Symbol*, 22:1–39.

A. Reining and B. Lönneker-Rodman. 2007. Corpus-driven metaphor harvesting. In *Proceedings of the HLT/NAACL-07 Workshop on Computational Approaches to Figurative Language*, pages 5–12, Rochester, New York.

P. Resnik. 1993. *Selection and Information: A Class-based Approach to Lexical Relationships*. Ph.D. thesis, Philadelphia, PA, USA.

E. Shutova and S. Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of LREC 2010*, Malta.

E. Shutova. 2010. Automatic metaphor interpretation as a paraphrasing task. In *Proceedings of NAACL 2010*, Los Angeles, USA.

G. J. Steen. 2007. Finding metaphor in discourse: Pragglejaz and beyond. *Cultura, Lenguaje y Representacion / Culture, Language and Representation (CLR), Revista de Estudios Culturales de la Universitat Jaume I*, 5:9–26.

A. Stefanowitsch. 2006. Corpus-based approaches to metaphor and metonymy. In A. Stefanowitsch and S. T. Gries, editors, *Corpus-Based Approaches to Metaphor and Metonymy*, Berlin. Mouton de Gruyter.

L. Sun and A. Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of EMNLP 2009*, pages 638–647, Singapore, August.

R. Tourangeau and R. Sternberg. 1982. Understanding and appreciating metaphors. *Cognition*, 11:203–244.

T. Veale and Y. Hao. 2008. A fluid knowledge representation for understanding and generating creative metaphors. In *Proceedings of COLING 2008*, pages 945–952, Manchester, UK.

A. M. Wallington, J. A. Barnden, P. Buchlovsky, L. Fellows, and S. R. Glasbey. 2003. Metaphor annotation: A systematic study. Technical report, School of Computer Science, The University of Birmingham.

Y. Wilks. 1975. A preferential pattern-seeking semantics for natural language inference. *Artificial Intelligence*, 6:53–74.

Y. Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11(3):197–223.

A Game-Theoretic Model of Metaphorical Bargaining

Beata Beigman Klebanov
Kellogg School of Management
Northwestern University
beata@northwestern.edu

Eyal Beigman
Washington University in St. Louis
beigman@wustl.edu

Abstract

We present a game-theoretic model of bargaining over a metaphor in the context of political communication, find its equilibrium, and use it to rationalize observed linguistic behavior. We argue that game theory is well suited for modeling discourse as a dynamic resulting from a number of conflicting pressures, and suggest applications of interest to computational linguists.

1 Introduction

A 13 Dec 1992 article in *The Times* starts thus:

> The European train chugged out of the station last night; for most of the day it looked as if it might be stalled there for some time. It managed to pull away at around 10:30 pm only after the Spanish prime minister, Felipe Gonzalez, forced the passengers in the first class carriages into a last minute whip round to sweeten the trip for the European Community's poor four: Spain, Portugal, Greece and Ireland.
>
> The fat controller, Helmut Kohl, beamed with satisfaction as the deal was done. The elegantly-suited Francois Mitterrand was equally satisfied. But nobody was as pleased as John Major, stationmaster for the UK presidency, for whom the agreement marked a scarce high point in a battered premiership.
>
> The departure had actually been delayed by seven months by Danes on the line. Just when that problem was solved, there was the voluble outbreak, orchestrated by Spain, from the poor four passengers demanding that they should travel free and be given spending money, too.
>
> The coupling of the carriages may not be reliably secure but the pan-European express is in motion. That few seem to agree the destination suggests that future arguments are inevitable at every set of points. Next stop: Copenhagen.

Apart from an entertaining read, the extended metaphor provides an elaborate conceptual correspondence between a familiar domain of train journeys and the unfolding process of European integration. Carriages are likened to nation states; passengers to their peoples; treaties to stations; politicians to responsible rail company employees.

In a compact form, the metaphor gives expression to both the small and the large scale of the process. It provides for the recent history: Denmark's failure to ratify the 1992 Maastricht treaty until opt-outs were negotiated later that year is compared to dissenters sabotaging the journey by laying on the tracks (*Danes on the line*); negotiations over the Cohesion Fund that would provide less developed regions with financial aid to help them comply with convergence criteria are likened to second class carriages with poor passengers for whom the journey had to be subsidized. At a more general level, the European integration is a purposeful movement towards some destination according to a worked out plan, getting safely through negotiation and implementation from one treaty to another, as a train moving on its rails through subsequent stations, with each nation being separate yet tied with everyone else. Numerous inferences regarding speed, timetables, stations, passengers, different classes of tickets, temporary obstacles on the tracks, and so on can be made by the reader based on the knowledge of train journeys, giving him or her a feeling of an enhanced understanding[1] of the highly complex process of European integration.

So apt was the metaphor that political fights were waged over its details (Musolff, 2000). Worries about destination were given an eloquent expression by Margaret Thatcher (*Sunday Times*, 20 Sept 1992):

> She warned EC leaders to stop their endless round of summits and take notice of their own people. "There is a fear that the European train will thunder forward, laden with its customary cargo of gravy, towards a destination neither wished for nor understood by electorates. But the train can be stopped," she said.

[1] More on enhanced understanding in sections 3.2 and 4.2.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 698–709,
Uppsala, Sweden, 11-16 July 2010.

The metaphor proved flexible enough for further elaboration. John Major, a Conservative PM of Britain, spoke on June 1st, 1994 about his vision of the decision making at the EU level, saying that he had never believed that Europe must act as one on every issue, and advocating "a sensible new approach, varying when it needs to, multi-track, multi-speed, multi-layered." He attempted to turn a largely negative Conservative take on the European train (see Thatcher above) into a tenable positive vision — each nation-carriage is now presumably a rather autonomous entity, waiting on a side track for the right locomotive, in a huge yet smoothly operating railroad system.

Major's political opponents offered their counter-frames. In both cases, the imagery of a large transportation system was taken up, yet turned around to suggest that "multi, for everyone" amounts to Britain being in "the slow lane," and a different image was suggested that makes the negative evaluation of Britain's opt-outs more poignant — a football metaphor, where relegation to the second division is a sign of a weak performance, and a school metaphor, where Britain is portrayed as an under-achiever:

John Cunningham, Labour He has admitted that his Government would let Britain fall behind in Europe. He is apparently willing to offer voluntary relegation to the second division in Europe, and he isn't even prepared to put up a fight. I believe that in any two-speed Europe, Britain must be up with those in the fast lane. Clearly Mr Major does not.

Paddy Ashdown, Liberal Democrat Are you really saying that the best that Britain can hope for under your leadership is ... the slow lane of a two-speed Europe? Most people in this country will want to aim higher, and will reject your view of a 'drop-out' Britain.

The pro-European camp rallied around the "Britain in the slow lane" version as a critical stance towards the government's European policy. Of the alternative metaphors, the school metaphor has some traction in the Euro discourse, where the European (mainly German) financial officers are compared to school authorities, and governments struggling to meet the strict convergence criteria to enter the Euro are compared to pupils that barely make the grade with Britain as a 'drop-out' who gave up even trying (Musolff, 2000).

The fact that European policy is being communicated and negotiated via a metaphor is not surprising; after all, "there is always someone willing to help us think by providing us with a metaphor that accords with HIS views."[2] From the point of view of the dynamics of political discourse, the puzzle is rather the apparent tendency of politicians to be compelled by the *rival's* metaphorical framework. Thatcher tries to turn the train metaphor used by the pro-EU camp around. Yet, assuming metaphors are matters of choice, why should Thatcher feel constrained by her rival's choice, why doesn't she ignore it and merely suggest a new metaphor of her own design? As the evidence above suggests, this is not Thatcher's idiosyncrasy, as Major and his rivals acted similarly. Can this dynamic be explained?

In this article, we use the explanatory framework of game theory, seeking to rationalize the observed behavior by designing a game that would produce, at equilibrium, the observed dynamics. Specifically, we formalize the notion that the price of "locking" the public into a metaphorical frame of reference is that a politician is coerced into staying within the metaphor as well, even if he or she is at the receiving end of a rival's rhetorical move.

Since the use of game theory is not common in computational linguistics, we first explain its main attributes, justify our decision to make use of it, and draw connections to research questions that can benefit from its application (section 2). Next, we design the game of bargaining over a metaphor, and find its equilibrium (section 3), followed by a discussion (section 4).

2 Game-Theoretic models

The basic construct is that of a game, that is, a model of participants in an interaction (called "players"), their goals (or "utilities") and allowable moves. Different moves yield different utilities for a player; it is assumed that each player would pick a strategy that maximizes her utility. The observable is the actual sequence of moves; importantly, these are assumed to be the optimal outcome (an equilibrium) of the relevant game. A popular notion of equilibrium is Nash equilibrium (Nash, 1950). For extensive form games (the type employed in this paper), the notion of *subgame perfect equilibirum* is typically used, denoting a Nash equilibrium that would remain such if the players start from any stage of the evolving game (Selten (1975; 1965)).

The task of a game theorist is to reverse-engineer the model for which the observed se-

[2] Capitalization in the original, Bolinger (1980, p. 146).

quence of actions is an equilibrium. The resulting model is thereby able to rationalize the observed behavior as a naturally emerging dynamics between agents maximizing certain utility functions. In economics, game-theoretic models are used to explain price change, organization of production, and market failures (Mas-Colell et al., 1995; von Neumann and Morgenstern, 1944); in biology — the operation of natural selection processes (Axelrod and Hamilton, 1981; Maynard Smith and Price, 1973); in social sciences — political institutions, collective action, and conflict (Greif, 2006; Schelling, 1997; North, 1990). In recent applications in linguistics, pragmatic phenoma such as implicatures are rendered as an equilibrium outcome of a communication game (Jäger and Ebert, 2008; van Rooij, 2008; Ross, 2007; van Rooij and Schulz, 2004; Parikh, 2001; Glazer and Rubinstein, 2001; Dekker and van Rooy, 2000).

Computing equilibria is simple for some games and quite evolved for others. For example, computing the equilibrium of a zero-sum game is equivalent to LP optimization (Luce and Raiffa, 1957); an equilibrium of general bimatrix games can be found using a pivoting algorithm (von Stengel, 2007; Lemke and Howson, 1964). Interesting connections have been pointed out between game theory and machine learning: Freund and Schapire (1996) present both online learning and boosting as a repeated zero-sum game; Shalev-Shwartz and Singer (2006) show similarly that loss minimization in online learning is akin to an equilibrium path in a repeated game.

While game theoretic models are not much utilized in computational linguistics, they are quite attractive to tackle some of the problems computational linguists are interested in. For example, generation of referring expressions (Paraboni et al., 2007; Gardent et al., 2004; Siddharthan and Copestake, 2004; Dale and Reiter, 1995) can be rendered as a communication game with utility functions that reflect pressures to use shorter expressions while avoiding excessive ambiguity (Clark and Parikh, 2007), with corpora annotated for entity mentions informing the design of a model. Generally, computational linguistics research produces algorithms to detect entities of various kinds, be it topics, named entities, metaphors, moves in a multi-party conversations, or syntactic constructions in large corpora; such primary data can be used to trace developments not only in chronological terms (Gruhl et al., 2004; Allan, 2002), but in strategic terms, i.e. in terms that reflect agendas of the actors, such as political agendas in legislatures (Quinn et al., 2006) or activist forums (Greene and Resnik, 2009), research agendas in group meetings (Morgan et al., 2001), or social agendas in speed-dates (Jurafsky et al., 2009). Game theoretical models are well suited for modeling dynamics that emerge under multiple, possibly conflicting constraints, as we exemplify in this article.

3 The model

We extend Rubinstein (1982) model of negotiation through offers and counter-offers between two players with a public benefit constraint.

The model consists of (1) two players representing the opposing sides, (2) a set of **frames** $X \subset \mathbb{R}^n$ compact and convex, (3) preference relations described by continuous utility functions $U_1, U_2 : X \to \mathbb{R}^+$, (4) a sequence of frames $\mathcal{X}_0 \subset \mathcal{X}_1 \ldots \subset 2^X$ that can be suggested to the public, and (5) a sequence of public preferences over frames in $\mathcal{X}_t$ for $t=0, 1, 2, \ldots$ described by a public utility function U_t^p.

The game proceeds as follows. Initially the frame is $F_0=X$. In odd rounds player 1 appeals to the public with a frame $A_t^1 \in \mathcal{X}_{t|_{F_t}}$, $\mathcal{X}_{t|_{F_t}} = \{A \in \mathcal{X}_t : A \subset F_t\}$, player 2 counters with a frame $A_t^2 \in \mathcal{X}_{t|_{F_t}}$. The public chooses one of the frames based on $U_t^p(A_t^i)$ with ties broken in 1's favor. The accepted frame becomes the current frame for the next round F_{t+1}. In even rounds the parts of players 1 and 2 are reversed.

A finite sequence $F_0, \ldots, F_{t-1}$ gives the *history* of the bargaining process up to t. A *strategy* σ_i of player i is a function specifying for any history $h=\{F_0, \ldots, F_{t-1}\}$ the move player i makes at time t, namely the frame A_t^i she chooses to address the public. A sequence $F_0, F_1, F_2, F_3, \ldots$ describes a path the bargaining process can take, leading to an *outcome* $\cap_{t=0}^{\infty} F_t$. The players' utility for an outcome is given by $U_i = \lim_{t\to\infty} \int_{F_t} U_i(x) d\chi_{F_t}$ for $i=1, 2$ where χ_{F_t} is a probability measure on F_t. If $\cap_{t=0}^{\infty} F_t = \{x\}$ the utility is the point utility of x otherwise it is the expected utility on the intersection set.

3.1 Player utility

For a given issue under discussion, such as European integration process, we order the possible

states of the world along a single dimension that spans the policy variations proposed by the different players (politicians). Politics of a single issue are routinely modeled as lying on a single dimension.[3] In the British context, various configurations of the unfolding European reality are situated along the line between high degree of integration and complete separatism; Liberal Democrats are the most pro-European party, while United Kingdom Independence Party are at the far-right end of the scale, preferring British withdrawal from the EU. The two major parties, Labour and Conservatives (Tories), prefer intermediate left-leaning and right-leaning positions, respectively. A schematic description is shown in figure 1.

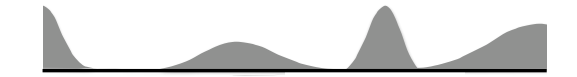


Figure 1: Preferences on pro-anti Europe axis.

The utilities of the different players can in this case be described as continuous single-peaked functions over an interval.[4] Thus X=$[0,1]$, and the utility functions $U_i(x)$=$\phi(||x-v_i||)$ for $v_i \in X$ where ϕ is a monotonically strictly decreasing function and $||\ ||$ is Euclidean distance.

3.2 Public utility

We note the difference between two types of utilities: The utility of the players is over outcomes, the utility of the public is over sets of outcomes (frames). The latter does not represent a utility the public has for one outcome or another, but rather a utility it has for an enhanced understanding. Thus, the public's utility from a frame is a function of the information content of the proposed frame relative to the current frame, i.e. the relative entropy of the two sets.[5] Formally, if the accepted frame at time t is F_t then for any Borel set $A \subset F_t$ the public utility for A is $U^p_t(A)$=$\Pi(\mathrm{Ent}_t(A))$ where $\mathrm{Ent}_t(A)$=$-\mu_t(A)\log\mu_t(A)$ for a continuous probability measure μ_t on F_t and Π is a continuous, monotone ascending function; for $A \not\subset F_t$, $U^p_t(A)$=0. We take μ_t to be the relative length of the segment $\mu_t(A)$=$\frac{|A|}{|F_t|}$, hence the entropy maximizing subsegments are of length $\frac{|F_t|}{2}$.

3.3 Game dynamics

At every point in the game, a certain set of the states-of-affairs is being deemed sufficiently probable by the public to require consideration. Suppose that initially any state of affairs within the interval $[0,1]$ is assigned a uniform probability and thus merits public attention. Each in her turn, the players propose to the public to concentrate on a subset of the currently considered states of affairs, arguing that those are the likelier ones to obtain, hence merit further attention. The metaphor used to deliver the proposal describes the newly proposed subset in a way that makes those states-of-affairs that are in it aligned with the metaphor, whereas all other states are left out of the proposed metaphorical frame. As the game proceeds, the public attention is concentrated on successively smaller sets of eventualities, and these are given a more and more detailed metaphoric description, providing the educational gratification of increasingly knowing better and better what is going on. At each step, each player strives to provide maximum public gratification while leading the public to focus on the frame (i.e. subset of states of affairs) that best meets the player's preferences.[6]

Figure 2 sketches the frame negotiation through train metaphor, from some point in time when the general train metaphor got established, through Thatcher's flashing out the issue of excessive speed and unclear direction, Major's multi-track corrective, and reply of his opponents on the left. The final frame has all those states of affairs that fit the extended metaphor – everyone is acting within the same broad system of rules, with Britain and perhaps others sometimes wanting to negotiate special, more gradual procedures, which would leave Britain less tightly integrated into the com-

[3] Indeed, Poole and Rosenthal (1997) argue that no more than two dimensions are needed to account for voting patterns on *all* issues in the US Congress.

[4] Single-peakedness is a common assumption in position modeling in political science (Downs, 1957).

[5] The notion that new beliefs are refinements of existing ones is current in contemporary theorizing about formation and change of beliefs, evaluations, and preferences. An update based on the latest available information is consistent with memory-based theories; in our model, in the equilibrium, the current frame contains information about the path-so-far, thus early stages of the bargaining processes are in some sense integrated into the current frame, compatible with the rival, online model of belief formation. See Druckman and Luria (2000) for a review of the relevant literature.

[6] We note that in our model every utterance has an impact on the public for which the player bears the consequences and is therefore a (costly) strategic move in the game. This is different from models of cheap talk such as Aumann (1990), Lewis (1969) where communication is devoid of strategic moves and is used primarily as a coordination device.

munity than some other European partners.

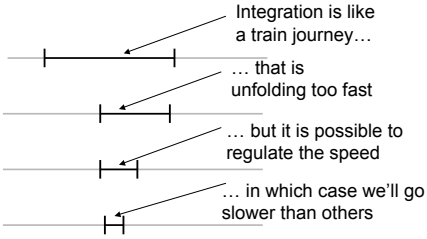


Figure 2: Bargaining over train metaphor.

3.4 The equilibrium

A pair of strategies (σ_1, σ_2) is a *Nash equilibrium* if there is no deviation strategy σ such that (σ, σ_2) leads to an outcome with higher utility for player 1 than outcome of (σ_1, σ_2) and the same for player 2. A *subgame* are all the possible moves following a history h=$\{F_0, \ldots, F_t\}$, in our case it is equivalent to a game with an initial frame F_t and the corresponding utilities. A *sub-strategy* is that part of the original strategy that is a strategy on the subgame. A pair of strategies is a *subgame perfect equilibrium* if, for any subgame, their sub-strategies are a Nash equilibrium.

Theorem 1 *In the frame bargaining game with single-peaked preferences*

1. *There exists a canonical subgame perfect equilibrium path* $F_0, F_1, F_2, \ldots$ *such that* $\cap_{t=0}^{\infty} F_t$=$\{x\}$.

2. *For any subgame perfect equilibrium path* $F'_0, F'_1, F'_2, \ldots$ *there exists* T *such that* $\cap_{t=0}^{\infty} F'_t$=$\cap_{t=0}^{T} F_t$.

The theorem states that the outcome of the bargaining will always be a frame on the canonical path. The rivals would suggest more specific frames either until convergence or until a situation where any further specification would produce a frame that "misses their point," so-to-speak, by removing too much of the favorable outcome space for both players. Figure 3 shows a situation where parties could decide to stall on the current frame: If player 1 has to choose between retaining F_0, or playing F_1 which would result in the rival's playing F_2, player 1 might choose to remain in F_0 if the utility of any outcome of the subgame starting from F_2 is lower than that of F_0, as long as player 1 believes that player 2 would reason similarly.

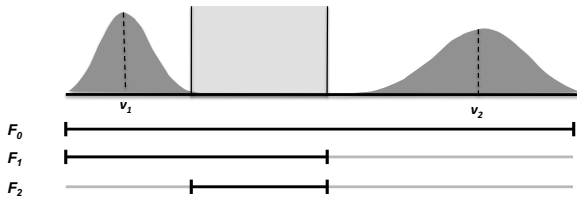


Figure 3: Stalled bargaining.

The idea of the proof is to construct a pair of strategies where each side attempts to pull the publicly accepted frame in the direction of its peak utility point. We show, assuming the peak of the first mover is to the left of peak of the second, that any deviation of the first mover would enable the second to shift the public frame more to the right, to an outcome of lower utility to the first mover. The full details of the proof of part 1 are given in the appendix; part 2 is proved in an accompanying technical report.

The equilibrium exhibits the following properties: (a) a first mover's advantage — for any player, the outcome would be closer to her peak point if she moves first than if she moves second; (b) a centrist's advantage — if a player moves first and her peak is closer to the middle of the initial frame, she can derive a higher utility from the outcome than if her peak were further from the middle. Please see appendix for justifications.

4 Discussion

4.1 Political communication

This article studies some properties of frame bargaining through metaphor in political communication, where rival politicians choose how to elaborate the current metaphor to educate the public about the ongoing situation in a way most consistent with their political preferences. Modeling the public preferences as highest relative entropy subset of possible states-of-affairs, we show that strategic choices by the politicians lead to a subgame perfect equilibrium where the less politically extreme player who moves first is at an advantage.

In a democracy, such player would typically be the government, as the bulk of voters do not by definition vote for extreme views, and since the government is the agent that brings about changes in the current states of affairs, and is thus the first and most prepared to explain them to the public. Indeed, Entman's model of frame activation in political discourse is hierarchical, with the govern-

ment (administration) being the topmost frame-activator, and opposition and media elites typically reacting to the administration's frame (Entman, 2003).

4.2 Metaphor in political communication

The role of metaphor in communication has long been a subject of interest, with views ranging from an ornament that beautifies the argument in the ancient rhetorical traditions, to the contemporary views of conceptual metaphor as permeating every aspect of life (Lakoff and Johnson, 1980).

In political communication specifically, metaphor has long been known as a framing device. Framing can be defined as "selecting and highlighting some facets of events or issues, and making connections among them in order to promote a particular interpretation, evaluation, or solution" (Entman, 2003). Metaphors are notorious for allowing subliminal framing, where the metaphor seems so natural that the aspects of the phenomenon in question that do *not* align with the metaphor are seamlessly concealed. For example, WAR AS A COMPETITIVE GAME metaphor emphasizes the glory of winning and the shame of defeat, but hides the death-and-suffering aspect of the war, which makes sports metaphors a strategic choice when wishing to arouse a pro-war sentiment in the audience (Lakoff, 1991). Such subliminal framing can often be effectively contested by merely exposing the frame.

Our examples show a different use of metaphor. Far from being subliminal or covert, the details of the metaphor, its implications, and the evaluation promoted by any given version are an important tool in the public discussion of a complex political issue. The function of metaphorical framing here resembles a pedagogical one, where rendering an abstract theory in physics (such as electricity) in concrete commonsensical terms (such as water flow) is an effective strategy to enhance the students' understanding of the former (Gentner and Gentner, 1983). The measure of success for a given version of the frame is its ability to sway the public in the evaluative direction envisioned by the author by providing sufficient *educational benefit*, so-to-speak, that is, convincingly rendering a good portion of a complex reality in accessible terms.

Once a frame is found that provides extensive education benefit, such as the EUROPEAN INTEGRATION AS TRAIN JOURNEY above, a politician's attempt to debunk a metaphor as inappropriate risk public antagonism, as this would be akin to taking the benefit of enhanced understanding away. Thus, rather than contesting the validity of the metaphoric frame, politicians strive to find a way to turn the metaphor around, i.e. accept the general framework, but focus on a previously unexplored aspect that would lead to a different evaluative tilt. Our results show that being the first to use an effective metaphor that manages to lock the public in its framework is a strategic advantage as the need to communicate with the same public would compel the rival to take up the metaphor of your choice. To our knowledge, this is the first explanation of the use of extended metaphor in political communication on a complex issue in terms of the agendas of the rival parties and the changing disposition of the public being addressed. It is an open question whether similar "locking in" of the public can be attained by non-metaphorical means, and whether the ensuing dynamics would be similar.

4.3 Social dynamics

This article contributes to the growing literature on modeling social linguistic behavior, like debates (Somasundaran and Wiebe, 2009), dating (Jurafsky et al., 2009; Ranganath et al., 2009), collaborative authoring and editing in wikis (Leuf and Cunningham, 2001) such as Wikipedia (Vuong et al., 2008; Kittur et al., 2007; Viégas et al., 2004). The latter literature in particular sees the social activity as an unfolding process, for example, detecting the onset and resolution of a controversy over the content of a Wikipedia article through tracking article talk[7] and deletion-and-reversion patterns. Somewhat similarly to the metaphor debate discussed in this article, Viégas et al. (2004) note first-mover advantage in Wikipedia authoring, that is, the first version gives the tone for the subsequent edits and has its parts survive for relatively many editing cycles. Finding out how the initial contribution constrains and guides subsequent edits of the content of a Wikipedia article and what kind of argumentative strategies are employed in persuading others to retain one's contribution is an interesting direction for future research.

A number of recent studies of the linguistic aspects of social processes are construed as if the

[7] a page separate from the main article that is devoted to the discussion of the edits

events are taking place all-at-once — there is no differentiation between early and later stages of a debate in Somasundaran and Wiebe (2009) or initial and subsequent speed-dates for the same subject in Jurafsky et al. (2009). Yet adopting a dynamic perspective stands to reason in such cases.

For example, Somasundaran and Wiebe (2009) built a system for recognizing stance in an online debate (such as pro-iPhone or pro-Blackberry on http://www.covinceme.net). They noticed that the task was complicated by concessions — acknowledgments of some virtues of the competitor before stating own preference. This is quite possibly an instance of debate dynamics whereby as the debate evolves certain common ground emerges between the sides and the focus of the debate changes from the initial stage of elucidating which features are better in which product to a stage where the "facts" are settled and acknowledged by both sides and the debate moves to evaluation of the relative importance of those features.

As another example, consider the construction of statistical models of various emotional and personality traits based on a corpus of speed dates such as Jurafsky et al. (2009). Take the trait of intelligence. In their experiment with speed-dates, Fisman et al. (2006) found that males tend to disprefer females they perceive as more intelligent or ambitious than themselves. Consequently, an intelligent female might choose to act less intelligent in later rounds of speed dating if she has not so far met a sufficiently intelligent male, assuming she prefers a less-intelligent male to no match at all.

Better sensitivity to the dynamics of social processes underlying the observed linguistic communication will we believe result in increased interest in game-theoretic models, as these are especially well suited to handle cases where the sides have certain goals and adapt their moves based on the current situations, the other side's move, and possibly other considerations, such as the need to address effectively a wider audience, beyond the specific interlocutors. A game theoretic explanation advances the understanding of the process being modeled, and hence of the applicability, and the potential adaptation, of statistical models developed on a certain dataset to situations that differ somewhat from the original data: For example, a corpus with more rounds of speed-dates per participant might suddenly make females seem smarter, or a debate with a longer history would feature more, and perhaps more elaborate, concessions.

5 Empirical challenges

We suggested that models of dynamics such as the one presented in this article be built over data where entities of interest are clearly identified. This article is based on chapters 1 and 2 of the book by Musolff (2000) which itself is informed by a corpus-linguistic analysis of metaphor in media discourse in Britain and Germany. We now discuss the state of affairs in empirical approaches to detecting metaphors.

5.1 Metaphors in NLP

Metaphors received increasing attention from computational linguistics community in the last two decades. The tasks that have been addressed are explication of the reasoning behind the metaphor (Barnden et al., 2002; Narayanan, 1999; Hobbs, 1992); detection of conventional metaphors between two specific domains (Mason, 2004); classification of words, phrases or sentences as metaphoric or non-metaphoric (Krishnakumaran and Zhu, 2007; Birke and Sarkar, 2006; Gedigian et al., 2006; Fass, 1991).

We are not aware of research on automatic methods specifically geared to recognition of extended metaphors. Indeed, most computational work cited above concentrates on the detection of a local incongruity due to a violation of selectional restrictions when the verb or one of its arguments is used metaphorically (as in *Protesters derailed the conference*). Extended metaphors are expected to be difficult for such approaches, since many of the clauses are completely situated in the source domain and hence no local incongruities exist (see examples on the first page of this article).

5.2 Data collection

Supervised approaches to metaphor detection need to rely on annotated data. While metaphors are ubiquitous in language, an annotation project that seeks to narrow the scope of relevant metaphors down to metaphors from a particular source domain (such as train journeys) that describe a particular target domain (such as European integration) and are uttered by certain entities (such as senior UK politicians) face the problem of sparsity of the relevant data in the larger discourse: A random sample of the size amenable to human an-

notation is unlikely to capture in sufficient detail material pertaining to the one metaphor of interest.

To increase the likelihood of finding mentions of the source domain, a lexicon of words from the source domain can be used to select documents (Hardie et al., 2007; Gedigian et al., 2006). Another approach is metaphor "harvesting" – hypothesizing that metaphors of interest would occur in close proximity to lexical items representing the target domain of the metaphor, such as the 4 word window around the lemma *Europe* used in Reining and Lönneker-Rodman (2007).

5.3 Data annotation

A further challenge is producing reliable annotations. Pragglejaz (2007) propose a methodology for testing metaphoricity of a word in discourse and report κ=0.56-0.70 agreement for a group of six highly expert annotators. Beigman Klebanov et al. (2008) report κ=0.66 for detecting paragraphs containing metaphors from the source domains LOVE and VEHICLE with multiple non-expert annotators, though other source domains that often feature highly conventionalized metaphors (like *structure* or *foundation* from BUILDLING domain) or are more abstract and difficult to delimit (such as AUTHORITY) present a more challenging annotation task.

5.4 Measuring metaphors

A fully empirical basis for the kind of model presented in this paper would also involve defining a metric on metaphors that would allow measuring the frame chosen by the given version of the metaphor relatively to other such frames – that is, quantifying which part of the "integration is a train journey" metaphor is covered by those states of affairs that also fit Thatcher's critical rendition.

6 Conclusion

This article addressed a specific communicative setting (rival politicians trying to "sell" to the public their versions of the unfolding realities and necessary policies) and a specific linguistic tool (an extended metaphor), showing that the particular use made of metaphor in such setting can be rationalized based on the characteristics of the setting.

Various questions now arise. Given the central role played by the public gratification constraint in our model, would conversational situations without the need to persuade the public, such as meetings of small groups of peers or phone conversations between friends, tend less to the use of extended metaphor? Conversely, does the use of extended metaphor in other settings testify to the existence of presumed onlookers who need to be "captured" in a particular version of reality — as in pedagogic or poetic context?

Considerations of the participants' agendas and their impact on the ensuing dynamics of the exchange would we believe lead to further interest in game theoretic models when addressing complex social dynamics in situations like collaborative authoring, debates, or dating, and will augment the existing mostly statistical approaches with a broader picture of the relevant communication.

A Proof of Existence of a Subgame Perfect Equilibrium

For a segment $[a, b]$ and $a \leq v_1 < v_2 \leq b$ let $U_1(x)=\phi(||x - v_1||)$ and $U_2(x)=\phi(||x - v_2||)$ be utility functions with peaks v_1 and v_2, respectively. For a history $h=\{F_0, \ldots, F_t\}$ where $F_t=[l_t, r_t]$, let $\sigma_1^*(h)$, player 1's move, be defined as choosing $F_{t+1}=[l_{t+1}, r_{t+1}]$ such that $|F_{t+1}|=\frac{|F_t|}{2}$, and r_{t+1} is as close as possible to v_1. σ_2^* sets l_{t+1} with respect to v_2 in a symmetric fashion. Since F_t shrinks by half every round, $\lim_{t\to\infty} l_t = \lim_{t\to\infty} r_t = x^*$, converging to a point. We now show (σ_1^*, σ_2^*) is an equilibrium by showing that neither player has a profitable deviation.

Notice that after the first round the subgame is identical to the initial game with F_1 replacing F_0, and the roles of players reversed. Player 2 had no influence on the choice of F_1, hence she has a profitable deviation iff she has a profitable deviation on the continuation subgame where she is the first mover. It thus suffices to show that the first mover (player 1) has no profitable deviations to establish that (σ_1^*, σ_2^*) is an equilibrium.

Since by definition σ_2^* always chooses an entropy maximizing segment, for player 1 to choose a non-entropy maximizing segment (more or less than half the length) amounts to yielding the round to player 2, which is equivalent in terms of the resulting accepted frame to a situation where player 1 chooses an entropy maximizing segment – the same one chosen by player 2. Thus we need to consider only deviations with entropy maximizing frames.

Step 1: Suppose σ_1' is a strategy of player 1 and let $F_0', F_1', F_2', \ldots$ be the sequence of frames on

the path corresponding to the pair (σ_1', σ_2^*). Let t_0 be the first move deviating from the equilibrium path, namely $F_{t_0} \neq F'_{t_0}$. We first show that F_{t_0-1} could not be (a) completely to the left of v_1 or (b) completely to the right of v_2. Suppose (a) holds. Then by definition $r_{t_0-2}=r_{t_0-1}<v_1$, and, inductively, $r_0=r_{t_0-1}<v_1$; this contradicts $r_0=1$ that follows from $F_0=[0,1]$. Possibility (b) is similarly refuted. Therefore, the only two cases for F_{t_0-1} with respect to v_1 are depicted in figure 4. Note that this implies $v_1 \leq x^* \leq v_2$.

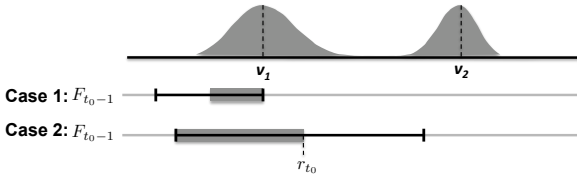


Figure 4: Two cases of current frame location.

Step 2: In case 1, σ_1^* will choose frames of type $[l_t, v_1]$ for any $t \geq t_0$, and σ_2^* will do the same on any history in the continuation game, hence the outcome will eventually be v_1. As this is player 1's peak utility point, she has no profitable deviation.

Step 3: In case 2, F_{t_0} is the leftmost entropy maximizing subsegment of F_{t_0-1} and the deviation F'_{t_0} can only be a shift to the right namely $r'_{t_0} \geq r_{t_0}$. If player 2 could choose $[v_2, r_{t_0+1}]$ given r_{t_0}, she can still choose the same frame given r'_{t_0}, so the outcome would be v_2 and F'_{t_0} was not profitable. If player 2 could not choose $[v_2, r_{t_0+1}]$ given r_{t_0}, implying that $x^* < v_2$, but as a result of the deviation can now choose $[v_2, r'_{t_0+1}]$, implying that the outcome would be v_2, clearly player 1 has not benefited from the deviation since U_1 is descending right of v_1. If player 2 still cannot choose $[v_2, r'_{t_0+1}]$ after the deviation, she would choose the rightmost entropy maximizing segment with $l'_{t_0+1} \geq l_{t_0+1}$. If this still allows player 1 to do $[l'_{t_0+2}, v_1]$ and hence to lead to v_1 as the outcome, it was possible in $[l_{t_0+2}, v_1]$ as well, so no profit is gained by having deviated. Otherwise, $r'_{t_0+2} \geq r_{t_0+2}$.

Step 3 can be repeated ad infinitum to show that $r'_t \geq r_t$ unless for some history h the deviation enables $\sigma_2(h)=[v_2, r'_t]$. In the former case we get $\lim_{t\to\infty} r'_t = x' \geq x^* = \lim_{t\to\infty} r_t$ where $\cap_{t=1}^{\infty} F'_t = \{x'\}$. Since r'_t and r_t are to the right of v_1 and U_1 is descending right of v_1 it follows that $U_1(x^*) \geq U_1(x')$. In the latter case $x' \geq v_2$. Since F_t is never strictly to the right of v_2, $x^* = \lim_{t\to\infty} l_t \leq v_2 \leq x'$, therefore $U_1(x^*) \geq U_1(x')$. In either case the deviation σ'_1 cannot result in a better outcome for player 1. This finishes the proof that (σ_1^*, σ_2^*) is a Nash equilibrium.

Notice that (σ_1^*, σ_2^*) prescribe sub-strategies on any subgame that are themselves Nash equilibria for the subgames, hence (σ_1^*, σ_2^*) is a subgame perfect equilibrium □

First Mover's Advantage: The proof of step 3 shows that having the left boundary of the current frame further to the right cannot yield a better outcome for player 1. Yet, if player 1's first turn comes after that of player 2, she will start with a current frame with the left boundary further to the right than the initial frame before player 2 moved, since moving the left boundary is player 2's equilibrium strategy. Hence a player would never achieve a better outcome starting second if both players are playing the canonical strategy.

Centrist's Advantage: Let M be the middle of F_0. Consider a more extreme version of player 1 — player $1^\#$. Suppose w.l.g. $v_1^\# < v_1 \leq M$. In case $v_1^\# < v_1 < v_2$, for all utilities u of the outcome of dynamics vs player 2, if player $1^\#$ could attain u, player 1 could attain u or more; the reverse is not true, for example when $|v_1^\# - l_t| < \frac{|F_t|}{2} \leq |v_1 - l_t|$ and player 1 (or $1^\#$) is moving first. In case $v_2 < v_1^\# < v_1$, if player 1 (or $1^\#$) moves first, she is able to force her peak point as the outcome. If $v_1^\# < v_2 < v_1$, player 1 can force v_1 as the outcome, whereas player $1^\#$ would not necessarily be able to force $v_1^\#$, as player 2 would pull the outcome towards v_2. Hence a first moving centrist is never worse off, and often better off, than a first moving extremist.

References

James Allan, editor. 2002. *Topic Detection and Tracking: Event-Based Information Organization*. Norwell, MA:Kluwer Academic Publishers.

Robert Aumann. 1990. Nash Equilibria are not Self-Enforcing. In Jean J. Gabszewicz, Jean-Francois Richard, and Laurence A. Wolsey, editors, *Economic Decision-Making: Games, Econometrics and Optimisation*, pages 201–206. Amsterdam: Elsevier.

Robert Axelrod and William D. Hamilton. 1981. The evolution of cooperation. *Science*, 211(4489):1390–1396.

John A. Barnden, Sheila R. Glasbey, Mark G. Lee, and Alan M. Wallington. 2002. Reasoning in metaphor

understanding: The ATT-Meta approach and system. In *Proceedings of COLING*, pages 121–128.

Beata Beigman Klebanov, Eyal Beigman, and Daniel Diermeier. 2008. Analyzing disagreements. In Ron Artstein, Gemma Boleda, Frank Keller, and Sabine Schulte im Walde, editors, *Proceedings of COLING Workshop on Human Judgments in Computational Linguistics*, pages 2–7, Manchester, UK, August. International Committee on Computational Linguistics.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of EACL*, pages 329–336.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Resarch*, 3:993–1022.

Dwight Bolinger. 1980. *Language – The Loaded Weapon*. London: Longman.

Robin Clark and Prashant Parikh. 2007. Game Theory and Discourse Anaphora. *Journal of Logic, Language and Information*, 16:265–282.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 18:233–263.

Paul Dekker and Robert van Rooy. 2000. Bidirectional optimality theory: An application of game theory. *Journal of Semantics*, 17(3):217–242.

Anthony Downs. 1957. An economic theory of political action in a democracy. *The Journal of Political Economy*, 65(2):135–150.

James Druckman and Arthur Luria. 2000. Preference formation. *Annual Review of Political Science*, 2:1–24.

Robert M. Entman. 2003. Cascading activation: Contesting the White House's frame after 9/11. *Political Communication*, 20:415–432.

Dan Fass. 1991. Met*: a method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

Raymond Fisman, Sheena Iyengar, Emir Kamenica, and Itamar Simonson. 2006. Gender Differences in Mate Selection: Evidence from a Speed Dating Experiment. *Quarterly Journal of Economics*, 121(2):673–697.

Yoav Freund and Robert E. Schapire. 1996. Game theory, on-line prediction, and boosting. In *Proceedings of the annual conference on Computational Learning Theory*, pages 325–332, Desenzano del Garda, Italy, June -July.

Claire Gardent, Hlne Manuélian, Kristina Striegnitz, and Marilisa Amoia. 2004. Generating Definite Descriptions: Non-Incrementality, Inference and Data. In Thomas Pechmann and Christopher Habel, editors, *Multidisciplinary Approaches to Language Production*. Mouton de Gruyter.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of NAACL Workshop on Scalable Natural Language Understanding*, pages 41–48.

Deidre Gentner and Donald Gentner. 1983. Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner and A. Stevens, editors, *Mental models*. Hillsdale, NJ: Lawrence Erlbaum.

Jacob Glazer and Ariel Rubinstein. 2001. Debates and decisions: On a rationale of argumentation rules. *Games and Economic Behavior*, 36(2):158–173.

Stephan Greene and Philip Resnik. 2009. More than Words: Syntactic Packaging and Implicit Sentiment. In *Proceedings of NAACL*, pages 503–511, Boulder, CO, June.

Avner Greif. 2006. *Institutions and the path to the modern economy: Lessons from medieval trade*. Cambridge University Press.

Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. 2004. Information diffusion through blogspace. In *Proceedings of the 13th international conference on World Wide Web*, pages 491–501.

Andrew Hardie, Veronika Koller, Paul Rayson, and Elena Semino. 2007. Exploiting a semantic annotation tool for metaphor analysis. In *Proceedings of the Corpus Linguistics Conference*, Birmingham, UK, Julyt.

Jerry Hobbs. 1992. Metaphor and abduction. In Andrew Ortony, Jon Slack, and Oliviero Stock, editors, *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues*, pages 35–58. Springer Verlag.

Gerhard Jäger and Christian Ebert. 2008. Pragmatic Rationalizability. In *Proceedings of the 13th annual meeting of Gesellschaft fur Semantik, Sinn und Bedeutung*, pages 1–15, Stuttgart, Germany, September-October.

Dan Jurafsky, Rajesh Ranganath, and Dan McFarland. 2009. Extracting social meaning: Identifying interactional style in spoken conversation. In *Proceedings of NAACL*, pages 638–646, Boulder, CO, June.

Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. 2007. He says, she says: Conflict and coordination in Wikipedia. In *CHI-07: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 453–462, San Jose, CA, USA.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of NAACL Workshop on Computational Approaches to Figurative Language*, pages 13–20.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. Chicago University Press.

George Lakoff. 1991. Metaphor and war: The metaphor system used to justify war in the Gulf. *Peace Research*, 23:25–32.

Carlton E. Lemke and Joseph T. Howson. 1964. Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

Bo Leuf and Ward Cunningham. 2001. *The Wiki way: quick collaboration on the Web*. Boston: Addison-Wesley.

David Lewis. 1969. *Convention*. Cambridge, MA: Harvard University Press.

Robert D. Luce and Howard Raiffa. 1957. *Games and decisions*. New York: John Wiley and Sons.

Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. 1995. *Microeconomic theory*. Oxford University Press.

Zachary J. Mason. 2004. CorMet: a computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44.

John Maynard Smith and George R. Price. 1973. The logic of animal conflict. *Nature*, 246(5427):15–18.

Nelson Morgan, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Adam Janin, Thilo Pfau, Elizabeth Shriberg, and Andreas Stolcke. 2001. The Meeting Project at ICSI. In *Proceedings of the HLT*, pages 246–252, San Diego, CA.

Andreas Musolff. 2000. *Mirror images of Europe: Metaphors in the public debate about Europe in Britain and Germany*. München: Iudicium.

Srini Narayanan. 1999. Moving right along: A computational model of metaphoric reasoning about events. In *Proceedings of AAAI*, pages 121–128.

John F. Nash. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49.

Douglass C. North. 1990. *Institutions, institutional change, and economic performance*. Cambridge University Press.

Ivandr Paraboni, Kees van Deemter, and Judith Masthoff. 2007. Generating Referring Expressions: Making Referents Easy to Identify. *Computational Lingusitics*, 33(2):229–254.

Prashant Parikh. 2001. *The Use of Language*. Stanford: CSLI Publications.

Keith T. Poole and Howard Rosenthal. 1997. *Congress: A Political-Economic History of Roll Call Voting*. Oxford University Press.

Group Pragglejaz. 2007. MIP: A Method for Identifying Metaphorically Used Words in Discourse. *Metaphor and Symbol*, 22(1):1–39.

Kevin M. Quinn, Burt L. Monroe, Michael Colaresi, Michael H. Crespin, and Dragomir R. Radev. 2006. An automated method of topic-coding legislative speech over time with application to the 105th-108th U.S. Senate. Unpublished Manuscript.

Rajesh Ranganath, Dan Jurafsky, and Dan McFarland. 2009. It's not you, it's me: Detecting flirting and its misperception in speed-dates. In *Proceedings of EMNLP*, pages 334–342, Singapore, August.

Astrid Reining and Birte Lönneker-Rodman. 2007. Corpus-driven metaphor harvesting. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 5–12, Rochester, New York.

Ian Ross. 2007. Situations and Solution Concepts in Game-Theoretic Approaches to Pragmatics. In Ahti-Veikko Pietarinen, editor, *Game Theory and Linguistic Meaning*, pages 135–147. Oxford, UK: Elsevier Ltd.

Ariel Rubinstein. 1982. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109.

Thomas C. Schelling. 1997. *The strategy of conflict*. Harvard University Press.

Reinhard Selten. 1965. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeitschrift für die Gesamte Staatswissenschaft*, 12:301–324.

Reinhard Selten. 1975. Re-examination of the Perfectness Concept for Equilibrium Points in Extensive Form Games. *International Journal of Game Theory*, 4:25–55.

Shai Shalev-Shwartz and Yoram Singer. 2006. Convex Repeated Games and Fenchel Duality. In *Proceedings of NIPS*, pages 1265–1272.

Advaith Siddharthan and Ann Copestake. 2004. Generating referring expressions in open domains. In *Proceedings of the ACL*, pages 407–414, Barcelona, Spain, July.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing Stances in Online Debates. In *Proceedings of the ACL*, pages 226–234.

Robert van Rooij and Katrin Schulz. 2004. Exhaustive Interpretation of Complex Sentences. *Journal of Logic, Language and Information*, 13(4):491–519.

Robert van Rooij. 2008. Games and Quantity implicatures. *Journal of Economic Methodology*, 15(3):261–274.

Fernanda B. Viégas, Martin Wattenberg, and Kushal Dave. 2004. Studying cooperation and conflict between authors with history flow visualizations. In *CHI-04: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 575–582, Vienna, Austria.

John von Neumann and Oskar Morgenstern. 1944. *Theory of games and economic behavior*. Princeton University Press.

Bernhard von Stengel. 2007. Equilibrium computation for two-player games in strategic and extensive form. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*, pages 53–78. Cambridge University Press.

Ba-Quy Vuong, Ee-Peng Lim, Aixin Sun, Minh-Tam Le, and Hady Wirawan Lauw. 2008. On ranking controversies in Wikipedia: models and evaluation. In *Proceedings of the international conference on Web Search and Web Data Mining*, pages 171–182, Palo Alto, CA, USA.

Kernel Based Discourse Relation Recognition with Temporal Ordering Information

WenTing Wang[1] **Jian Su**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research
1 Fusionopolis Way, #21-01 Connexis
Singapore 138632
{wwang,sujian}@i2r.a-star.edu.sg

[2]Department of Computer Science
University of Singapore
Singapore 117417
tacl@comp.nus.edu.sg

Abstract

Syntactic knowledge is important for discourse relation recognition. Yet only heuristically selected flat paths and 2-level production rules have been used to incorporate such information so far. In this paper we propose using tree kernel based approach to automatically mine the syntactic information from the parse trees for discourse analysis, applying kernel function to the tree structures directly. These structural syntactic features, together with other normal flat features are incorporated into our composite kernel to capture diverse knowledge for simultaneous discourse identification and classification for both explicit and implicit relations. The experiment shows tree kernel approach is able to give statistical significant improvements over flat syntactic path feature. We also illustrate that tree kernel approach covers more structure information than the production rules, which allows tree kernel to further incorporate information from a higher dimension space for possible better discrimination. Besides, we further propose to leverage on temporal ordering information to constrain the interpretation of discourse relation, which also demonstrate statistical significant improvements for discourse relation recognition on PDTB 2.0 for both explicit and implicit as well.

1 Introduction

Discourse relations capture the internal structure and logical relationship of coherent text, including *Temporal*, *Causal* and *Contrastive* relations etc. The ability of recognizing such relations between text units including identifying and classifying provides important information to other natural language processing systems, such as language generation, document summarization, and question answering. For example, *Causal* relation can be used to answer more sophisticated, non-factoid *'Why'* questions.

Lee et al. (2006) demonstrates that modeling discourse structure requires prior linguistic analysis on syntax. This shows the importance of syntactic knowledge to discourse analysis. However, most of previous work only deploys lexical and semantic features (Marcu and Echihabi, 2002; Pettibone and PonBarry, 2003; Saito et al., 2006; Ben and James, 2007; Lin et al., 2009; Pitler et al., 2009) with only two exceptions (Ben and James, 2007; Lin et al., 2009). Nevertheless, Ben and James (2007) only uses flat syntactic path connecting connective and arguments in the parse tree. The hierarchical structured information in the trees is not well preserved in their flat syntactic path features. Besides, such a syntactic feature selected and defined according to linguistic intuition has its limitation, as it remains unclear what kinds of syntactic heuristics are effective for discourse analysis.

The more recent work from Lin et al. (2009) uses 2-level production rules to represent parse tree information. Yet it doesn't cover all the other sub-trees structural information which can be also useful for the recognition.

In this paper we propose using tree kernel based method to automatically mine the syntactic

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 710–719,
Uppsala, Sweden, 11-16 July 2010.

information from the parse trees for discourse analysis, applying kernel function to the parse tree structures directly. These structural syntactic features, together with other flat features are then incorporated into our composite kernel to capture diverse knowledge for simultaneous discourse identification and classification. The experiment shows that tree kernel is able to effectively incorporate syntactic structural information and produce statistical significant improvements over flat syntactic path feature for the recognition of both explicit and implicit relation in Penn Discourse Treebank (PDTB; Prasad et al., 2008). We also illustrate that tree kernel approach covers more structure information than the production rules, which allows tree kernel to further work on a higher dimensional space for possible better discrimination.

Besides, inspired by the linguistic study on tense and discourse anaphor (Webber, 1988), we further propose to incorporate temporal ordering information to constrain the interpretation of discourse relation, which also demonstrates statistical significant improvements for discourse relation recognition on PDTB v2.0 for both explicit and implicit relations.

The organization of the rest of the paper is as follows. We briefly introduce PDTB in Section 2. Section 3 gives the related work on tree kernel approach in NLP and its difference with production rules, and also linguistic study on tense and discourse anaphor. Section 4 introduces the frame work for discourse recognition, as well as the baseline feature space and the SVM classifier. We present our kernel-based method in Section 5, and the usage of temporal ordering feature in Section 6. Section 7 shows the experiments and discussions. We conclude our works in Section 8.

2 Penn Discourse Tree Bank

The Penn Discourse Treebank (PDTB) is the largest available annotated corpora of discourse relations (Prasad et al., 2008) over 2,312 Wall Street Journal articles. The PDTB models discourse relation in the predicate-argument view, where a discourse connective (e.g., *but*) is treated as a predicate taking two text spans as its arguments. The argument that the discourse connective syntactically bounds to is called Arg2, and the other argument is called Arg1.

The PDTB provides annotations for both explicit and implicit discourse relations. An *explicit* relation is triggered by an explicit connective. Example (1) shows an explicit *Contrast* relation signaled by the discourse connective *'but'*.

(1). Arg1. *Yesterday, the retailing and financial services giant reported a 16% drop in third-quarter earnings to $257.5 million, or 75 cents a share, from a restated $305 million, or 80 cents a share, a year earlier.*
Arg2. <u>But</u> the news was even worse for Sears's core U.S. retailing operation, the largest in the nation.

In the PDTB, *local* implicit relations are also annotated. The annotators insert a connective expression that best conveys the inferred *implicit* relation between adjacent sentences within the same paragraph. In Example (2), the annotators select *'because'* as the most appropriate connective to express the inferred *Causal* relation between the sentences. There is one special label *AltLex* pre-defined for cases where the insertion of an *Implicit* connective to express an inferred relation led to a *redundancy* in the expression of the relation. In Example (3), the *Causal* relation derived between sentences is *alternatively lexicalized* by some *non-connective expression* shown in square brackets, so no *implicit* connective is inserted. In our experiments, we treat *AltLex* Relations the same way as normal *Implicit* relations.

(2). Arg1. *Some have raised their cash positions to record levels.*
Arg2. <u>Implicit = Because</u> High cash positions help buffer a fund when the market falls.

(3). Arg1. *Ms. Bartlett's previous work, which earned her an international reputation in the non-horticultural art world, often took gardens as its nominal subject.*
Arg2. [Mayhap this metaphorical connection made] the BPC Fine Arts Committee think she had a literal green thumb.

The PDTB also captures two non-implicit cases: (a) *Entity* relation where the relation between adjacent sentences is based on entity coherence (Knott et al., 2001) as in Example (4); and (b) *No* relation where no discourse or entity-based coherence relation can be inferred between adjacent sentences.

(4). But for South Garden, the grid was to be a 3-D network of masonry or hedge walls with real plants inside them.

In a Letter to the BPCA, kelly/varnell called *this* "arbitrary and amateurish."

Each *Explicit*, *Implicit* and *AltLex* relation is annotated with a sense. The *senses* in PDTB are arranged in a three-level hierarchy. The top level has four tags representing four major semantic classes: *Temporal*, *Contingency*, *Comparison* and *Expansion*. For each class, a second level of *types* is defined to further refine the semantic of the class levels. For example, *Contingency* has two types *Cause* and *Condition*. A third level of *subtype* specifies the semantic contribution of each argument. In our experiments, we use only the top level of the sense annotations.

3 Related Work

Tree Kernel based Approach in NLP. While the feature based approach may not be able to fully utilize the syntactic information in a parse tree, an alternative to the feature-based methods, tree kernel methods (Haussler, 1999) have been proposed to implicitly explore features in a high dimensional space by employing a kernel function to calculate the similarity between two objects directly. In particular, the kernel methods could be very effective at reducing the burden of feature engineering for structured objects in NLP research (Culotta and Sorensen, 2004). This is because a kernel can measure the similarity between two discrete structured objects by directly using the original representation of the objects instead of explicitly enumerating their features.

Indeed, using kernel methods to mine structural knowledge has shown success in some NLP applications like parsing (Collins and Duffy, 2001; Moschitti, 2004) and relation extraction (Zelenko et al., 2003; Zhang et al., 2006). However, to our knowledge, the application of such a technique to discourse relation recognition still remains unexplored.

Lin et al. (2009) has explored the 2-level production rules for discourse analysis. However, Figure 1 shows that only 2-level sub-tree structures (e.g. T_a - T_e) are covered in production rules. Other sub-trees beyond 2-level (e.g. T_f - T_j) are only captured in the tree kernel, which allows tree kernel to further leverage on information from higher dimension space for possible better discrimination. Especially, when there are enough training data, this is similar to the study

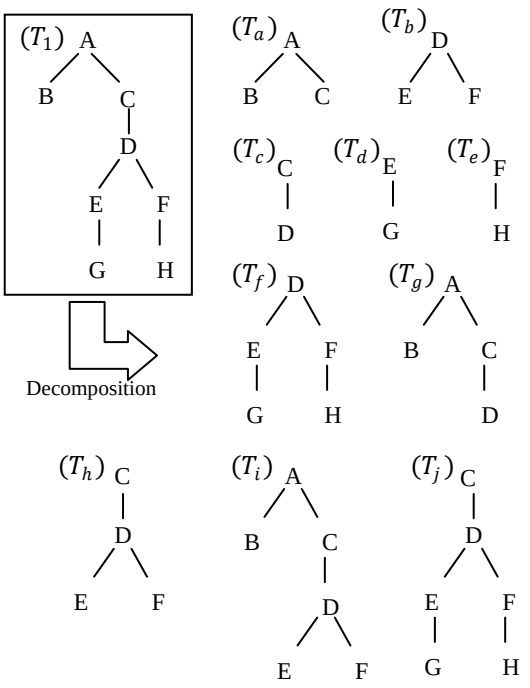


Figure 1. Different sub-tree sets for T_1 used by 2-level production rules and convolution tree kernel approaches. T_a-T_j and T_1 itself are covered by tree kernel, while only T_a-T_e are covered by production rules.

on language modeling that N-gram beyond unigram and bigram further improves the performance in large corpus.

Tense and Temporal Ordering Information. Linguistic studies (Webber, 1988) show that a tensed clause C_b provides two pieces of semantic information: (a) a description of an event (or situation) E_b; and (b) a particular configuration of the *point of event* (ET), the *point of reference* (RT) and the *point of speech* (ST). Both the characteristics of E_b and the configuration of ET, RT and ST are critical to interpret the relationship of event E_b with other events in the discourse model. Our observation on temporal ordering information is in line with the above, which is also incorporated in our discourse analyzer.

4 The Recognition Framework

In the learning framework, a training or testing instance is formed by a non-overlapping clause(s)/sentence(s) pair. Specifically, since *implicit* relations in PDTB are defined to be *local*, only clauses from adjacent sentences are paired for implicit cases. During training, for each discourse relation encountered, a positive instance is created by pairing the two arguments. Also a

set of negative instances is formed by paring each argument with neighboring non-argument clauses or sentences. Based on the training instances, a binary classifier is generated for each type using a particular learning algorithm. During resolution, (a) clauses within same sentence and sentences within three-sentence spans are paired to form an explicit testing instance; and (b) neighboring sentences within three-sentence spans are paired to form an implicit testing instance. The instance is presented to each explicit or implicit relation classifier which then returns a class label with a confidence value indicating the likelihood that the candidate pair holds a particular discourse relation. The relation with the highest confidence value will be assigned to the pair.

4.1 Base Features

In our system, the base features adopted include lexical pair, distance and attribution etc. as listed in Table 1. All these base features have been proved effective for discourse analysis in previous work.

Feature Names	Description
(F1)	cue phrase
(F2)	neighboring punctuation
(F3)	position of connective if presents
(F4)	extents of arguments
(F5)	relative order of arguments
(F6)	distance between arguments
(F7)	grammatical role of arguments
(F8)	lexical pairs
(F9)	attribution

Table 1. Base Feature Set

4.2 Support Vector Machine

In theory, any discriminative learning algorithm is applicable to learn the classifier for discourse analysis. In our study, we use Support Vector Machine (Vapnik, 1995) to allow the use of kernels to incorporate the structure feature.

Suppose the training set S consists of labeled vectors $\{(x_i, y_i)\}$, where x_i is the feature vector of a training instance and y_i is its class label. The classifier learned by SVM is:

$$f(x) = sgn\left(\sum_{i=1} y_i a_i x * x_i + b\right)$$

where a_i is the learned parameter for a feature vector x_i, and b is another parameter which can be derived from a_i . A testing instance x is classified as positive if $f(x) > 0$[1].

One advantage of SVM is that we can use tree kernel approach to capture syntactic parse tree information in a particular high-dimension space.

In the next section, we will discuss how to use kernel to incorporate the more complex structure feature.

5 Incorporating Structural Syntactic Information

A parse tree that covers both discourse arguments could provide us much syntactic information related to the pair. Both the syntactic flat path connecting connective and arguments and the 2-level production rules in the parse tree used in previous study can be directly described by the tree structure. Other syntactic knowledge that may be helpful for discourse resolution could also be implicitly represented in the tree. Therefore, by comparing the common sub-structures between two trees we can find out to which level two trees contain similar syntactic information, which can be done using a convolution tree kernel.

The value returned from the tree kernel reflects the similarity between two instances in syntax. Such syntactic similarity can be further combined with other flat linguistic features to compute the overall similarity between two instances through a composite kernel. And thus an SVM classifier can be learned and then used for recognition.

5.1 Structural Syntactic Feature

Parsing is a sentence level processing. However, in many cases two discourse arguments do not occur in the same sentence. To present their syntactic properties and relations in a single tree structure, we construct a syntax tree for each paragraph by attaching the parsing trees of all its sentences to an upper paragraph node. In this paper, we only consider discourse relations within 3 sentences, which only occur within each pa-

[1] In our task, the result of $f(x)$ is used as the confidence value of the candidate argument pair x to hold a particular discourse relation.

ragraph, thus paragraph parse trees are sufficient. Our 3-sentence spans cover 95% discourse relation cases in PDTB v2.0.

Having obtained the parse tree of a paragraph, we shall consider how to select the appropriate portion of the tree as the structured feature for a given instance. As each instance is related to two arguments, the structured feature at least should be able to cover both of these two arguments. Generally, the more substructure of the tree is included, the more syntactic information would be provided, but at the same time the more noisy information would likely be introduced. In our study, we examine three structured features that contain different substructures of the paragraph parse tree:

Min-Expansion This feature records the minimal structure covering both arguments and connective word in the parse tree. It only includes the nodes occurring in the shortest path connecting Arg1, Arg2 and connective, via the nearest commonly commanding node. For example, considering Example (5), Figure 2 illustrates the representation of the structured feature for this relation instance. Note that the two clauses underlined with dashed lines are *attributions* which are not part of the relation.

(5). Arg1. *Suppression of the book,* Judge Oakes observed, *would operate as a prior restraint and thus involve the First Amendment.*

Arg2. Moreover, and here Judge Oakes went to the heart of the question, "Responsible biographers and historians constantly use primary sources, letters, diaries and memoranda."

Simple-Expansion *Min-Expansion* could, to some degree, describe the syntactic relationships between the connective and arguments. However, the syntactic properties of the argument pair might not be captured, because the tree structure surrounding the argument is not taken into consideration. To incorporate such information, *Simple-Expansion* not only contains all the nodes in *Min-Expansion*, but also includes the first-level children of these nodes[2]. Figure 3 illustrates such a feature for Example (5). We can see that the nodes "PRN" in both sentences are included in the feature.

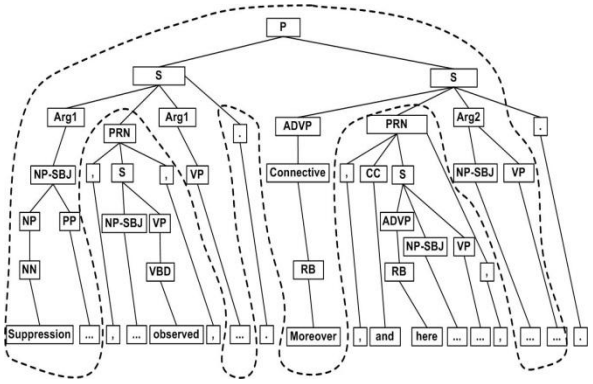


Figure 2. Min-Expansion tree built from golden standard parse tree for the explicit discourse relation in Example (5). Note that to distinguish from other words, we explicitly mark up in the structured feature the arguments and connective, by appending a string tag "Arg1", "Arg2" and "Connective" respectively.

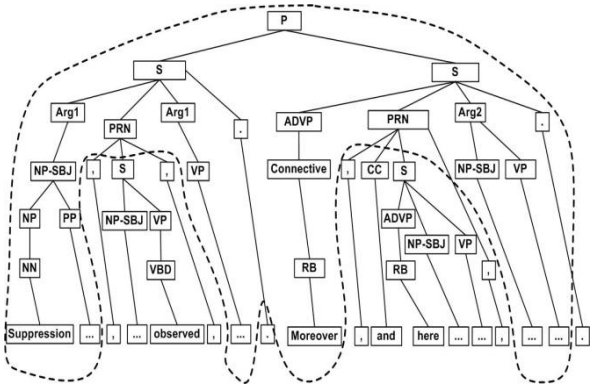


Figure 3. Simple-Expansion tree for the explicit discourse relation in Example (5).

Full-Expansion This feature focuses on the tree structure between two arguments. It not only includes all the nodes in *Simple-Expansion*, but also the nodes (beneath the nearest commanding parent) that cover the words between the two arguments. Such a feature keeps the most information related to the argument pair. Figure 4

[2] We will not expand the nodes denoting the sentences other than where the arguments occur.

shows the structure for feature *Full-Expansion* of Example (5). As illustrated, different from in *Simple-Expansion*, each sub-tree of "PRN" in each sentence is fully expanded and all its children nodes are included in *Full-Expansion*.

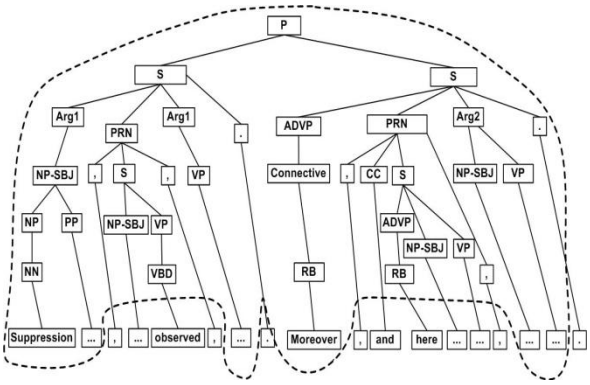


Figure 4. Full-Expansion tree for the explicit discourse relation in Example (5).

5.2 Convolution Parse Tree Kernel

Given the parse tree defined above, we use the same convolution tree kernel as described in (Collins and Duffy, 2002) and (Moschitti, 2004). In general, we can represent a parse tree T by a vector of integer counts of each sub-tree type (regardless of its ancestors):

$$\emptyset(T) = (\#of\ subtrees\ of\ type\ 1, \dots, \#\ of\ subtrees\ of type\ I, \dots, \#\ of\ subtrees\ of\ type\ n).$$

This results in a very high dimensionality since the number of different sub-trees is exponential in its size. Thus, it is computational infeasible to directly use the feature vector $\emptyset(T)$. To solve the computational issue, a tree kernel function is introduced to calculate the dot product between the above high dimensional vectors efficiently.

Given two tree segments T_1 and T_2, the tree kernel function is defined:

$$\begin{aligned} K(T_1, T_2) &= < \emptyset(T_1), \emptyset(T_2) > \\ &= \sum_i \emptyset(T_1)[i], \emptyset(T_2)[i] \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) * I_i(n_2) \end{aligned}$$

where N_1 and N_2 are the sets of all nodes in trees T_1 and T_2, respectively; and $I_i(n)$ is the indicator function that is 1 iff a subtree of type i occurs with root at node n or zero otherwise. (Collins and Duffy, 2002) shows that $K(T_1, T_2)$ is an instance of convolution kernels over tree structures, and can be computed in $O(|N_1|, |N_2|)$ by the following recursive definitions:

$$\Delta(n_1, n_2) = \sum_i I_i(n_1) * I_i(n_2)$$

(1) $\Delta(n_1, n_2) = 0$ if n_1 and n_2 do not have the same syntactic tag or their children are different;

(2) else if both n_1 and n_2 are pre-terminals (i.e. POS tags), $\Delta(n_1, n_2) = 1 \times \lambda$;

(3) else, $\Delta(n_1, n_2) =$

$$\lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j))),$$

where $nc(n_1)$ is the number of the children of n_1, $ch(n, j)$ is the j^{th} child of node n and λ ($0 < \lambda < 1$) is the decay factor in order to make the kernel value less variable with respect to the sub-tree sizes. In addition, the recursive rule (3) holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offspring.

The parse tree kernel counts the number of common sub-trees as the syntactic similarity measure between two instances. The time complexity for computing this kernel is $O(|N_1| \cdot |N_2|)$.

5.3 Composite Tree Kernel

Besides the above convolution parse tree kernel $\widehat{K}_{tree}(x_1, x_2) = K(T_1, T_2)$ defined to capture the syntactic information between two instances x_1 and x_2, we also use another kernel $\widehat{K}_{flat}$ to capture other flat features, such as base features (described in Table 1) and temporal ordering information (described in Section 6). In our study, the composite kernel is defined in the following way:

$$\begin{aligned} \widehat{K}_1(x_1, x_2) = \alpha \cdot \widehat{K}_{flat}(x_1, x_2) + \\ (1 - \alpha) \cdot \widehat{K}_{tree}(x_1, x_2). \end{aligned}$$

Here, $\widehat{K}(\cdot, \cdot)$ can be normalized by $\widehat{K}(y, z) = K(y, z)/\sqrt{K(y, y) \cdot K(z, z)}$ and α is the coefficient.

6 Using Temporal Ordering Information

In our discourse analyzer, we also add in temporal information to be used as features to predict discourse relations. This is because both our observations and some linguistic studies (Webber, 1988) show that temporal ordering information including tense, aspectual and event orders between two arguments may constrain the discourse relation type. For example, the connective

word is the same in both Example (6) and (7), but the tense shift from progressive form in clause 6.a to simple past form in clause 6.b, indicating that the *twisting* occurred during the state of *running the marathon*, usually signals a *temporal* discourse relation; while in Example (7), both clauses are in past tense and it is marked as a *Causal* relation.

(6). a. Yesterday Holly was running a marathon
b. ***when*** she twisted her ankle.

(7). a. Use of dispersants was approved
b. ***when*** a test on the third day showed some positive results.

Inspired by the linguistic model from Webber (1988) as described in Section 3, we explore the temporal order of events in two adjacent sentences for discourse relation interpretation. Here event is represented by the *head of verb*, and the temporal order refers to the *logical* occurrence (i.e. before/at/after) between events. For instance, the event ordering in Example (8) can be interpreted as:

$$Event(broken) \prec_{before} Event(went) .$$

8. a. John went to the hospital.
b. He had broken his ankle on a patch of ice.

We notice that the feasible temporal order of events differs for different discourse relations. For example, in *causal* relations, *cause* event usually happens before *effect* event, i.e.

$$Event(cause) \prec_{before} Event(effect).$$

So it is possible to infer a *causal* relation in Example (8) if and only if 8.b is taken to be the *cause* event and 8.a is taken to be the effect event. That is, 8.b is taken as happening prior to *his going into hospital*.

In our experiments, we use the TARSQI[3] system to identify event, analyze tense and aspectual information, and label the temporal order of events. Then the tense and temporal ordering information is extracted as features for discourse relation recognition.

[3] http://www.isi.edu/tarsqi/

7 Experiments and Results

In this section we provide the results of a set of experiments focused on the task of simultaneous discourse identification and classification.

7.1 Experimental Settings

We experiment on PDTB v2.0 corpus. Besides four top-level discourse relations, we also consider *Entity* and *No* relations described in Section 2. We directly use the golden standard parse trees in Penn TreeBank. We employ an SVM coreference resolver trained and tested on ACE 2005 with 79.5% Precision, 66.7% Recall and 72.5% F_1 to label coreference mentions of the same named entity in an article. For learning, we use the binary SVMLight developed by (Joachims, 1998) and Tree Kernel Toolkits developed by (Moschitti, 2004). All classifiers are trained with default learning parameters.

The performance is evaluated using *Accuracy* which is calculated as follow:

$$Accuracy = \frac{TruePositive + TrueNegative}{All}$$

Sections 2-22 are used for training and Sections 23-24 for testing. In this paper, we only consider any non-overlapping clauses/sentences pair in 3-sentence spans. For training, there were 14812, 12843 and 4410 instances for *Explicit*, *Implicit* and *Entity+No* relations respectively; while for testing, the number was 1489, 1167 and 380.

7.2 System with Structural Kernel

Table 2 lists the performance of simultaneous identification and classification on level-1 discourse senses. In the first row, only base features described in Section 4 are used. In the second row, we test Ben and James (2007)'s algorithm which uses heuristically defined syntactic paths and acts as a good baseline to compare with our learned-based approach using the structured information. The last three rows of Table 2 reports the results combining base features with three syntactic structured features (i.e. *Min-Expansion*, *Simple-Expansion* and *Full-Expansion*) described in Section 5.

We can see that all our tree kernels outperform the manually constructed flat path feature in all three groups including *Explicit* only, *Implicit* only and *All* relations, with the accuracy increasing by 1.8%, 6.7% and 3.1% respectively. Especially, it shows that structural syntactic information is more helpful for *Implicit* cases which is generally much harder than *Explicit* cases. We

Features	Accuracy		
	Explicit	Implicit	All
Base Features	67.1	29	48.6
Base + Manually selected flat path features	70.3	32	52.6
Base + Tree kernel (Min-Expansion)	71.9	38.6	55.6
Base + Tree kernel (Simple-Expansion)	**72.1**	**38.7**	**55.7**
Base + Tree kernel (Full-Expansion)	71.8	38.4	55.4

Table 2. Results of the syntactic structured kernels on level-1 discourse relation recognition.

conduct chi square statistical significance test on *All* relations between flat path approach and *Simple-Expansion* approach, which shows the performance improvements are statistical significant ($\rho < 0.05$) through incorporating tree kernel. This proves that structural syntactic information has good predication power for discourse analysis in both explicit and implicit relations. We also observe that among the three syntactic structured features, *Min-Expansion* and *Simple-Expansion* achieve similar performances which are better than the result for *Full-Expansion*. This may be due to that most significant information is with the arguments and the shortest path connecting connectives and arguments. However, *Full-Expansion* that includes more information in other branches may introduce too many details which are rather tangential to discourse recognition. Our subsequent reports will focus on *Simple-Expansion*, unless otherwise specified.

As described in Section 5, to compute the structural information, parse trees for different sentences are connected to form a large tree for a paragraph. It would be interesting to find how the structured information works for discourse relations whose arguments reside in different sentences. For this purpose, we test the accuracy for discourse relations with the two arguments occurring in the same sentence, one-sentence apart, and two-sentence apart. Table 3 compares the learning systems with/without the structured feature present. From the table, for all three cases, the accuracies drop with the increase of the distances between the two arguments. However, adding the structured information would bring consistent improvement against the baselines regardless of the number of sentence distance. This observation suggests that the structured syntactic information is more helpful for inter-sentential discourse analysis.

We also concern about how the structured information works for identification and classification respectively. Table 4 lists the results for the two sub-tasks. As shown, with the structured information incorporated, the system (Base + Tree Kernel) can boost the performance of the two baselines (Base Features in the first row andBase + Manually selected paths in the second row), for both identification and classification respectively. We also observe that the structural syntactic information is more helpful for classification task which is generally harder than identification. This is in line with the intuition that classification is generally a much harder task. We find that due to the weak modeling of *Entity* relations, many *Entity* relations which are non-discourse relation instances are mis-identified as implicit *Expansion* relations. Nevertheless, it clearly directs our future work.

Sentence Distance	0 (959)	1 (1746)	2 (331)
Base Features	52	49.2	35.5
Base + Manually selected flat path features	56.7	52	43.8
Base + Tree Kernel	58.3	55.6	49.7

Table 3. Results of the syntactic structured kernel for discourse relations recognition with arguments in different sentences apart.

Tasks	Identification	Classification
Base Features	58.6	50.5
Base + Manually selected flat path features	59.7	52.6
Base + Tree Kernel	63.3	59.3

Table 4. Results of the syntactic structured kernel for simultaneous discourse identification and classification subtasks.

7.3 System with Temporal Ordering Information

To examine the effectiveness of our temporal ordering information, we perform experiments

on simultaneous identification and classification of level-1 discourse relations to compare with using only base feature set as baseline. The results are shown in Table 5. We observe that the use of temporal ordering information increases the accuracy by 3%, 3.6% and 3.2% for *Explicit*, *Implicit* and *All* groups respectively. We conduct chi square statistical significant test on *All* relations, which shows the performance improvement is statistical significant ($\rho < 0.05$). It indicates that temporal ordering information can constrain the discourse relation types inferred within a clause(s)/sentence(s) pair for both explicit and implicit relations.

Features	Accuracy		
	Explicit	Implicit	All
Base Features	67.1	29	48.6
Base + Temporal Ordering Information	70.1	32.6	51.8

Table 5. Results of tense and temporal order information on level-1 discourse relations.

We observe that although temporal ordering information is useful in both explicit and implicit relation recognition, the contributions of the specific information are quite different for the two cases. In our experiments, we use tense and aspectual information for explicit relations, while event ordering information is used for implicit relations. The reason is explicit connective itself provides a strong hint for explicit relation, so tense and aspectual analysis which yields a reliable result can provide additional constraints, thus can help explicit relation recognition. However, event ordering which would inevitably involve more noises will adversely affect the explicit relation recognition performance. On the other hand, for implicit relations with no explicit connective words, tense and aspectual information alone is not enough for discourse analysis. Event ordering can provide more necessary information to further constrain the inferred relations.

7.4 Overall Results

We also evaluate our model which combines base features, tree kernel and tense/temporal ordering information together on *Explicit*, *Implicit* and *All* Relations respectively. The overall results are shown in Table 6.

Relations	Accuracy
Explicit	74.2
Implicit	40.0
All	57.3

Table 6. Overall results for combined model (Base + Tree Kernel + Tense/Temporal).

8 Conclusions and Future Works

The purpose of this paper is to explore how to make use of the structural syntactic knowledge to do discourse relation recognition. In previous work, syntactic information from parse trees is represented as a set of heuristically selected flat paths or 2-level production rules. However, the features defined this way may not necessarily capture all useful syntactic information provided by the parse trees for discourse analysis. In the paper, we propose a kernel-based method to incorporate the structural information embedded in parse trees. Specifically, we directly utilize the syntactic parse tree as a structure feature, and then apply kernels to such a feature, together with other normal features. The experimental results on PDTB v2.0 show that our kernel-based approach is able to give statistical significant improvement over flat syntactic path method. In addition, we also propose to incorporate temporal ordering information to constrain the interpretation of discourse relations, which also demonstrate statistical significant improvements for discourse relation recognition, both explicit and implicit.

In future, we plan to model *Entity* relations which constitute 24% of *Implicit+Entity+No* relation cases, thus to improve the accuracy of relation detection.

Reference

Ben W. and James P. 2007. *Automatically Identifying the Arguments of Discourse Connectives*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 92-101.

Culotta A. and Sorensen J. 2004. *Dependency Tree Kernel for Relation Extraction*. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2004), pages 423-429.

Collins M. and Duffy N. 2001. *New Ranking Algorithms for Parsing and Tagging: Kernels over Dis-*

crete Structures and the Voted Perceptron. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), pages 263-270.

Collins M. and Duffy N. 2002. *Convolution Kernels for Natural Language.* NIPS-2001.

Haussler D. 1999. *Convolution Kernels on Discrete Structures*. Technical Report UCS-CRL-99-10, University of California, Santa Cruz.

Joachims T. 1999. *Making Large-scale SVM Learning Practical*. In Advances in Kernel Methods – Support Vector Learning. MIT Press.

Knott, A., Oberlander, J., O'Donnel, M., and Mellish, C. 2001. *Beyond elaboration: the interaction of relations and focus in coherent text.* In T. Sanders, J. Schilperoord, and W. Spooren, editors, Text Representation: Linguistic and Psycholinguistics Aspects, pages 181-196. Benjamins, Amsterdam.

Lee A., Prasad R., Joshi A., Dinesh N. and Webber B. 2006. *Complexity of dependencies in discourse: are dependencies in discourse more complex than in syntax?* In Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories. Prague, Czech Republic, December.

Lin Z., Kan M. and Ng H. 2009. *Recognizing Implicit Discourse Relations in the Penn Discourse Treebank*. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), Singapore, August.

Marcu D. and Echihabi A. 2002. *An Unsupervised Approach to Recognizing Discourse Relations*. In Proceedings of the 40th Annual Meeting of ACL, pages 368-375.

Moschitti A. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. In Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics (ACL 2004), pages 335-342.

Pettibone J. and Pon-Barry H. 2003. *A Maximum Entropy Approach to Recognizing Discourse Relations in Spoken Language*. Working Paper. The Stanford Natural Language Processing Group, June 6.

Pitler E., Louis A. and Nenkova A. 2009. *Automatic Sense Predication for Implicit Discourse Relations in Text*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009).

Prasad R., Dinesh N., Lee A., Miltsakaki E., Robaldo L., Joshi A. and Webber B. 2008. T*he Penn Discourse TreeBank 2.0*. In Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008).

Saito M., Yamamoto K. and Sekine S. 2006. *Using phrasal patterns to identify discourse relations*. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006), pages 133–136, New York, USA.

Vapnik V. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Webber Bonnie. 1988. *Tense as Discourse Anaphor*. Computational Linguistics, 14:61–73.

Zelenko D., Aone C. and Richardella A. 2003. *Kernel Methods for Relation Extraction*. Journal of Machine Learning Research, 3(6):1083-1106.

Zhang M., Zhang J. and Su J. *Exploring Syntactic Features for Relation Extraction using a Convolution Tree Kernel*. In Proceedings of the Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL 2006), New York, USA.

Hierarchical Joint Learning: Improving Joint Parsing and Named Entity Recognition with Non-Jointly Labeled Data

Jenny Rose Finkel and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

Abstract

One of the main obstacles to producing high quality joint models is the lack of jointly annotated data. Joint modeling of multiple natural language processing tasks outperforms single-task models learned from the same data, but still underperforms compared to single-task models learned on the more abundant quantities of available single-task annotated data. In this paper we present a novel model which makes use of additional single-task annotated data to improve the performance of a joint model. Our model utilizes a hierarchical prior to link the feature weights for shared features in several single-task models and the joint model. Experiments on joint parsing and named entity recognition, using the OntoNotes corpus, show that our hierarchical joint model can produce substantial gains over a joint model trained on only the jointly annotated data.

1 Introduction

Joint learning of multiple types of linguistic structure results in models which produce more consistent outputs, and for which performance improves across all aspects of the joint structure. Joint models can be particularly useful for producing analyses of sentences which are used as input for higher-level, more semantically-oriented systems, such as question answering and machine translation. These high-level systems typically combine the outputs from many low-level systems, such as parsing, named entity recognition (NER) and coreference resolution. When trained separately, these single-task models can produce outputs which are inconsistent with one another, such as named entities which do not correspond to any nodes in the parse tree (see Figure 1 for an example). Moreover, one expects that the different types of annotations should provide useful information to one another, and that modeling them jointly should improve performance. Because a named entity should correspond to a node in the parse tree, strong evidence about either aspect of the model should positively impact the other aspect.

However, designing joint models which actually improve performance has proven challenging. The CoNLL 2008 shared task (Surdeanu et al., 2008) was on joint parsing and semantic role labeling, but the best systems (Johansson and Nugues, 2008) were the ones which completely decoupled the tasks. While negative results are rarely published, this was not the first failed attempt at joint parsing and semantic role labeling (Sutton and McCallum, 2005). There have been some recent successes with joint modeling. Zhang and Clark (2008) built a perceptron-based joint segmenter and part-of-speech (POS) tagger for Chinese, and Toutanova and Cherry (2009) learned a joint model of lemmatization and POS tagging which outperformed a pipelined model. Adler and Elhadad (2006) presented an HMM-based approach for unsupervised joint morphological segmentation and tagging of Hebrew, and Goldberg and Tsarfaty (2008) developed a joint model of segmentation, tagging and parsing of Hebrew, based on lattice parsing. No discussion of joint modeling would be complete without mention of (Miller et al., 2000), who trained a Collins-style generative parser (Collins, 1997) over a syntactic structure augmented with the *template entity* and *template relations* annotations for the MUC-7 shared task.

One significant limitation for many joint models is the lack of jointly annotated data. We built a joint model of parsing and named entity recognition (Finkel and Manning, 2009b), which had small gains on parse performance and moderate gains on named entity performance, when compared with single-task models trained on the same data. However, the performance of our model, trained using the OntoNotes corpus (Hovy et al., 2006), fell short of separate parsing and named

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 720–728,
Uppsala, Sweden, 11-16 July 2010.

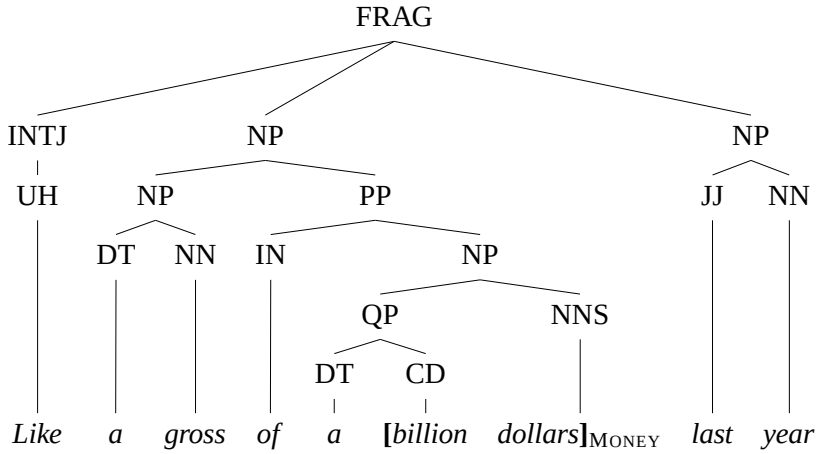


Figure 1: Example from the data where separate parse and named entity models give conflicting output.

entity models trained on larger corpora, annotated with only one type of information.

This paper addresses the problem of how to learn high-quality joint models with smaller quantities of jointly-annotated data that has been augmented with larger amounts of single-task annotated data. To our knowledge this work is the first attempt at such a task. We use a hierarchical prior to link a joint model trained on jointly-annotated data with other single-task models trained on single-task annotated data. The key to making this work is for the joint model to share some features with each of the single-task models. Then, the singly-annotated data can be used to influence the feature weights for the shared features in the joint model. This is an important contribution, because it provides all the benefits of joint modeling, but without the high cost of jointly annotating large corpora. We applied our hierarchical joint model to parsing and named entity recognition, and it reduced errors by over 20% on both tasks when compared to a joint model trained on only the jointly annotated data.

2 Related Work

Our task can be viewed as an instance of *multi-task learning*, a machine learning paradigm in which the objective is to simultaneously solve multiple, related tasks for which you have separate labeled training data. Many schemes for multitask learning, including the one we use here, are instances of hierarchical models. There has not been much work on multi-task learning in the NLP community; Daumé III (2007) and Finkel and Manning (2009a) both build models for multi-domain learning, a variant on domain adaptation where there exists labeled training data for all domains and the goal is to improve performance on all of them. Ando and Zhang (2005) utilized a multi-task learner within their semi-supervised algorithm to learn feature representations which were useful across a large number of related tasks. Outside of the NLP community, Elidan et al. (2008) used an undirected Bayesian transfer hierarchy to jointly model the shapes of multiple mammal species. Evgeniou et al. (2005) applied a hierarchical prior to modeling exam scores of students. Other instances of multi-task learning include (Baxter, 1997; Caruana, 1997; Yu et al., 2005; Xue et al., 2007). For a more general discussion of hierarchical models, we direct the reader to Chapter 5 of (Gelman et al., 2003) and Chapter 12 of (Gelman and Hill, 2006).

3 Hierarchical Joint Learning

In this section we will discuss the main contribution of this paper, our hierarchical joint model which improves joint modeling performance through the use of *single-task models* which can be trained on *singly-annotated data*. Our experiments are on a joint parsing and named entity task, but the technique is more general and only requires that the *base models* (the joint model and single-task models) share some features. This section covers the general technique, and we will cover the details of the parsing, named entity, and joint models that we use in Section 4.

3.1 Intuitive Overview

As discussed, we have a joint model which requires jointly-annotated data, and several single-task models which only require singly-annotated data. The key to our hierarchical model is that the joint model must have features in common with each of the single models, though it can also have features which are only present in the joint model.

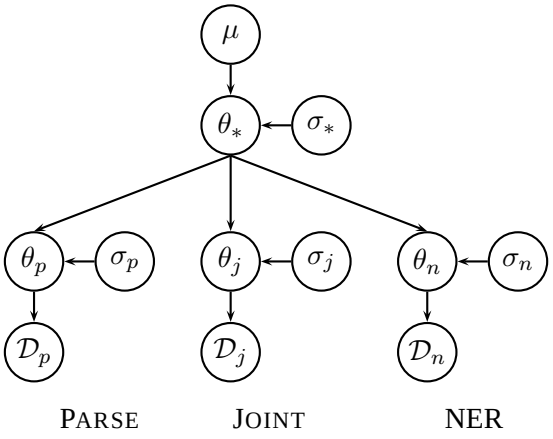


Figure 2: A graphical representation of our hierarchical joint model. There are separate base models for just parsing, just NER, and joint parsing and NER. The parameters for these models are linked via a hierarchical prior.

Each model has its own set of parameters (feature weights). However, parameters for the features which are shared between the single-task models and the joint model are able to influence one another via a hierarchical prior. This prior encourages the learned weights for the different models to be similar to one another. After training has been completed, we retain only the joint model's parameters. Our resulting joint model is of higher quality than a comparable joint model trained on only the jointly-annotated data, due to all of the evidence provided by the additional single-task data.

3.2 Formal Model

We have a set $\mathcal{M}$ of three base models: a parse-only model, an NER-only model and a joint model. These have corresponding log-likelihood functions $\mathcal{L}_p(\mathcal{D}_p;\theta_p)$, $\mathcal{L}_n(\mathcal{D}_n;\theta_n)$, and $\mathcal{L}_j(\mathcal{D}_j;\theta_j)$, where the $\mathcal{D}$s are the training data for each model, and the θs are the model-specific parameter (feature weight) vectors. These likelihood functions do *not* include priors over the θs. For representational simplicity, we assume that each of these vectors is the same size and corresponds to the same ordering of features. Features which don't apply to a particular model type (e.g., parse features in the named entity model) will always be zero, so their weights have no impact on that model's likelihood function. Conversely, allowing the presence of those features in models for which they do not apply will not influence their weights in the other models because there will be no evidence about them in the data. These three models are linked by a hierarchical prior, and their feature weight vectors are all drawn from this prior. The parameters θ_* for this prior have the same dimensionality as the model-specific parameters θ_m and are drawn from another, top-level prior. In our case, this top-level prior is a zero-mean Gaussian.[1]

The graphical representation of our hierarchical model is shown in Figure 2. The log-likelihood of this model is

$$\mathcal{L}_{\textit{hier-joint}}(\mathcal{D};\theta) = \sum_{m\in\mathcal{M}} \left(\mathcal{L}_m(\mathcal{D}_m;\theta_m) - \sum_i \frac{(\theta_{m,i}-\theta_{*,i})^2}{2\sigma_m^2} \right) - \sum_i \frac{(\theta_{*,i}-\mu_i)^2}{2\sigma_*^2} \quad (1)$$

The first summation in this equation computes the log-likelihood of each model, using the data and parameters which correspond to that model, and the prior likelihood of that model's parameters, based on a Gaussian prior centered around the top-level, non-model-specific parameters θ_*, and with model-specific variance σ_m. The final summation in the equation computes the prior likelihood of the top-level parameters θ_* according to a Gaussian prior with variance σ_* and mean μ (typically zero). This formulation encourages each base model to have feature weights similar to the top-level parameters (and hence one another).

The effects of the variances σ_m and σ_* warrant some discussion. σ_* has the familiar interpretation of dictating how much the model "cares" about feature weights diverging from zero (or μ). The model-specific variances, σ_m, have an entirely different interpretation. They dictate how how strong the penalty is for the domain-specific parameters to diverge from one another (via their similarity to θ_*). When σ_m are very low, then they are encouraged to be very similar, and taken to the extreme this is equivalent to completely tying the parameters between the tasks. When σ_m are very high, then there is less encouragement for the parameters to be similar, and taken to the extreme this is equivalent to completely decoupling the tasks.

We need to compute partial derivatives in order to optimize the model parameters. The partial derivatives for the parameters for each base model m are given by:

$$\frac{\partial \mathcal{L}_{\textit{hier}}(\mathcal{D};\theta)}{\partial \theta_{m,i}} = \frac{\partial \mathcal{L}_m(\mathcal{D}_m,\theta_m)}{\partial \theta_{m,i}} - \frac{\theta_{m,i}-\theta_{*,i}}{\sigma_d^2} \quad (2)$$

where the first term is the partial derivative according to the base model, and the second term is

[1] Though we use a zero-mean Gaussian prior, this top-level prior could take many forms, including an L_1 prior, or another hierarchical prior.

the prior centered around the top-level parameters. The partial derivatives for the top level parameters θ_* are:

$$\frac{\partial \mathcal{L}_{\textit{hier}}(\mathcal{D};\theta)}{\partial \theta_{*,i}} = \left(\sum_{m \in \mathcal{M}} \frac{\theta_{*,i} - \theta_{m,i}}{\sigma_m^2} \right) - \frac{\theta_{*,i} - \mu_i}{\sigma_*^2} \tag{3}$$

where the first term relates to how far each model-specific weight vector is from the top-level parameter values, and the second term relates how far each top-level parameter is from zero.

When a model has strong evidence for a feature, effectively what happens is that it pulls the value of the top-level parameter for that feature closer to the model-specific value for it. When it has little or no evidence for a feature then it will be pulled in the direction of the top-level parameter for that feature, whose value was influenced by the models which have evidence for that feature.

3.3 Optimization with Stochastic Gradient Descent

Inference in joint models tends to be slow, and often requires the use of stochastic optimization in order for the optimization to be tractable. L-BFGS and gradient descent, two frequently used numerical optimization algorithms, require computing the value and partial derivatives of the objective function using the entire training set. Instead, we use stochastic gradient descent. It requires a stochastic objective function, which is meant to be a low computational cost estimate of the real objective function. In most NLP models, such as logistic regression with a Gaussian prior, computing the stochastic objective function is fairly straightforward: you compute the model likelihood and partial derivatives for a randomly sampled subset of the training data. When computing the term for the prior, it must be rescaled by multiplying its value and derivatives by the proportion of the training data used. The stochastic objective function, where $\widehat{\mathcal{D}} \subseteq \mathcal{D}$ is a randomly drawn subset of the full training set, is given by

$$\mathcal{L}_{stoch}(\mathcal{D};\theta) = \mathcal{L}_{orig}(\widehat{\mathcal{D}};\theta) - \frac{|\widehat{\mathcal{D}}|}{|\mathcal{D}|} \sum_i \frac{(\theta_{*,i})^2}{2\sigma_*^2} \tag{4}$$

This is a *stochastic* function, and multiple calls to it with the same $\mathcal{D}$ and θ will produce different values because $\widehat{\mathcal{D}}$ is resampled each time. When designing a stochastic objective function, the critical fact to keep in mind is that the summed values and partial derivatives for any split of the data need to be equal to that of the full dataset. In practice, stochastic gradient descent only makes use of the partial derivatives and not the function value, so we will focus the remainder of the discussion on how to rescale the partial derivatives.

We now describe the more complicated case of stochastic optimization with a hierarchical objective function. For the sake of simplicity, let us assume that we are using a batch size of one, meaning $|\widehat{\mathcal{D}}| = 1$ in the above equation. Note that in the hierarchical model, each datum (sentence) in each base model should be weighted equally, so whichever dataset is the largest should be proportionally more likely to have one of its data sampled. For the sampled datum d, we then compute the function value and partial derivatives with respect to the correct base model for that datum. When we rescale the model-specific prior, we rescale based on the number of data in that model's training set, *not* the total number of data in all the models combined. Having uniformly randomly drawn datum $d \in \bigcup_{m \in \mathcal{M}} \mathcal{D}_m$, let $m(d) \in \mathcal{M}$ tell us to which model's training data the datum belongs. The stochastic partial derivatives will equal zero for all model parameters θ_m such that $m \neq m(d)$, and for $\theta_{m(d)}$ it becomes:

$$\frac{\partial \mathcal{L}_{\textit{hier-stoch}}(\mathcal{D};\theta)}{\partial \theta_{m(d),i}} = \tag{5}$$
$$\frac{\partial \mathcal{L}_{m(d)}(\{d\};\theta_{m(d)})}{\partial \theta_{m(d),i}} - \frac{1}{|\mathcal{D}_{m(d)}|} \left(\frac{\theta_{m(d),i} - \theta_{*,i}}{\sigma_d^2} \right)$$

Now we will discuss the stochastic partial derivatives with respect to the top-level parameters θ_*, which requires modifying Equation 3. The first term in that equation is a summation over all the models. In the stochastic derivative we only perform this computation for the datum's model $m(d)$, and then we rescale that value based on the number of data in that datum's model $|\mathcal{D}_{m(d)}|$. The second term in that equation is rescaled by the *total* number of data in all models combined. The stochastic partial derivatives with respect to θ_* become:

$$\frac{\partial \mathcal{L}_{\textit{hier-stoch}}(\mathcal{D};\theta)}{\partial \theta_{*,i}} = \tag{6}$$
$$\frac{1}{|\mathcal{D}_{m(d)}|} \left(\frac{\theta_{*,i} - \theta_{m(d),i}}{\sigma_m^2} \right) - \frac{1}{\sum_{m \in \mathcal{M}} |\mathcal{D}_m|} \left(\frac{\theta_{*,i}}{\sigma_*^2} \right)$$

where for conciseness we omit μ under the assumption that it equals zero.

An equally correct formulation for the partial derivative of θ_* is to simply rescale Equation 3 by the *total* number of data in all models. Early experiments found that both versions gave similar performance, but the latter was significantly

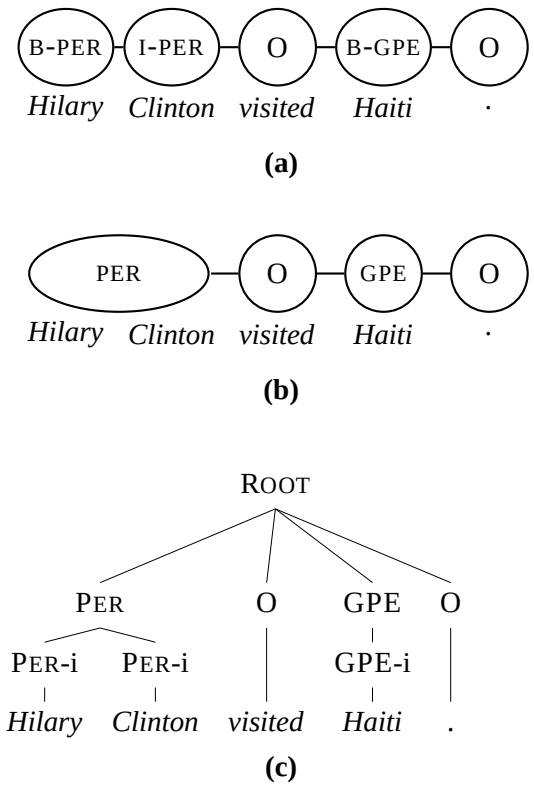


Figure 3: A linear-chain CRF **(a)** labels each word, whereas a semi-CRF **(b)** labels entire entities. A semi-CRF can be represented as a tree **(c)**, where **i** indicates an internal node for an entity.

slower to compute because it required summing over the parameter vectors for all base models instead of just the vector for the datum's model.

When using a batch size larger than one, you compute the given functions for each datum in the batch and then add them together.

4 Base Models

Our hierarchical joint model is composed of three separate models, one for just named entity recognition, one for just parsing, and one for joint parsing and named entity recognition. In this section we will review each of these models individually.

4.1 Semi-CRF for Named Entity Recognition

For our named entity recognition model we use a semi-CRF (Sarawagi and Cohen, 2004; Andrew, 2006). Semi-CRFs are very similar to the more popular linear-chain CRFs, but with several key advantages. Semi-CRFs *segment and label* the text simultaneously, whereas a linear-chain CRF will *only label* each word, and segmentation is implied by the labels assigned to the words. When doing named entity recognition, a semi-CRF will have one node for each entity, unlike a regular CRF which will have one node for each word.[2] See Figure 3a-b for an example of a semi-CRF and a linear-chain CRF over the same sentence. Note that the entity *Hilary Clinton* has one node in the semi-CRF representation, but two nodes in the linear-chain CRF. Because different segmentations have different model structures in a semi-CRF, one has to consider all possible structures (segmentations) as well as all possible labelings. It is common practice to limit segment length in order to speed up inference, as this allows for the use of a modified version of the forward-backward algorithm. When segment length is not restricted, the inference procedure is the same as that used in parsing (Finkel and Manning, 2009c).[3] In this work we do not enforce a length restriction, and directly utilize the fact that the model can be transformed into a parsing model. Figure 3c shows a parse tree representation of a semi-CRF.

While a linear-chain CRF allows features over adjacent words, a semi-CRF allows them over adjacent segments. This means that a semi-CRF can utilize all features used by a linear-chain CRF, and can also utilize features over entire segments, such as *First National Bank of New York City*, instead of just adjacent words like *First National* and *Bank of*. Let $\mathbf{y}$ be a vector representing the labeling for an entire sentence. y_i encodes the label of the ith segment, along with the span of words the segment encompasses. Let θ be the feature weights, and $\mathbf{f}(s, y_i, y_{i-1})$ the feature function over adjacent segments y_i and y_{i-1} in sentence s.[4] The log likelihood of a semi-CRF for a single sentence s is given by:

$$\mathcal{L}(\mathbf{y}|s;\theta) = \frac{1}{Z_s} \sum_{i=1}^{|\mathbf{y}|} \exp\{\theta \cdot \mathbf{f}(s, y_i, y_{i-1})\} \quad (7)$$

The partition function Z_s serves as a normalizer. It requires summing over the set $\mathbf{y}_s$ of all possible segmentations and labelings for the sentence s:

$$Z_s = \sum_{\mathbf{y} \in \mathbf{y}_s} \sum_{i=1}^{|\mathbf{y}|} \exp\{\theta \cdot \mathbf{f}(s, y_i, y_{i-1})\} \quad (8)$$

[2] Both models will have one node per word for non-entity words.

[3] While converting a semi-CRF into a parser results in much slower inference than a linear-chain CRF, it is still significantly faster than a treebank parser due to the reduced number of labels.

[4] There can also be features over single entities, but these can be encoded in the feature function over adjacent entities, so for notational simplicity we do not include an additional term for them.

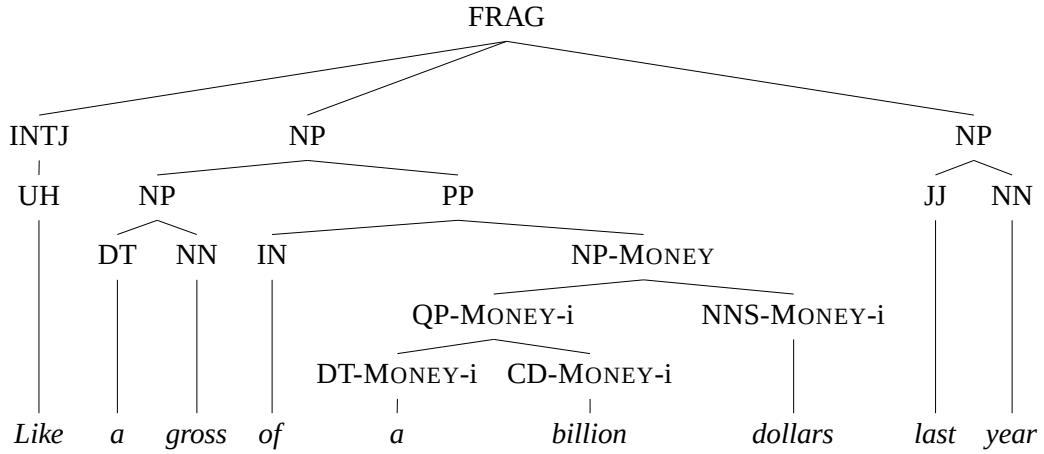


Figure 4: An example of a sentence jointly annotated with parse and named entity information. Named entities correspond to nodes in the tree, and the parse label is augmented with the named entity information.

Because we use a tree representation, it is easy to ensure that the features used in the NER model are identical to those in the joint parsing and named entity model, because the joint model (which we will discuss in Section 4.3) is also based on a tree representation where each entity corresponds to a single node in the tree.

4.2 CRF-CFG for Parsing

Our parsing model is the discriminatively trained, conditional random field-based context-free grammar parser (CRF-CFG) of (Finkel et al., 2008). The relationship between a CRF-CFG and a PCFG is analogous to the relationship between a linear-chain CRF and a hidden Markov model (HMM) for modeling sequence data. Let t be a complete parse tree for sentence s, and each local subtree $r \in t$ encodes both the rule from the grammar, and the span and split information (e.g $\text{NP}_{(7,9)} \rightarrow \text{JJ}_{(7,8)}\text{NN}_{(8,9)}$ which covers the last two words in Figure 1). The feature function $\mathbf{f}(r, s)$ computes the features, which are defined over a local subtree r and the words of the sentence. Let θ be the vector of feature weights. The log-likelihood of tree t over sentence s is:

$$\mathcal{L}(t|s;\theta) = \frac{1}{Z_s} \sum_{r \in t} \exp\{\theta \cdot \mathbf{f}(r, s)\} \quad (9)$$

To compute the partition function Z_s, which serves to normalize the function, we must sum over $\tau(s)$, the set of all possible parse trees for sentence s. The partition function is given by:

$$Z_s = \sum_{t' \in \tau(s)} \sum_{r \in t'} \exp\{\theta \cdot \mathbf{f}(r, s)\}$$

We also need to compute the partial derivatives which are used during optimization. Let $f_i(r, s)$ be the value of feature i for subtree r over sentence s, and let $E_\theta[f_i|s]$ be the expected value of feature i in sentence s, based on the current model parameters θ. The partial derivatives of θ are then given by

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_{(t,s) \in \mathcal{D}} \left(\left(\sum_{r \in t} f_i(r, s) \right) - E_\theta[f_i|s] \right) \quad (10)$$

Just like with a linear-chain CRF, this equation will be zero when the feature expectations in the model equal the feature values in the training data.

A variant of the inside-outside algorithm is used to efficiently compute the likelihood and partial derivatives. See (Finkel et al., 2008) for details.

4.3 Joint Model of Parsing and Named Entity Recognition

Our base joint model for parsing and named entity recognition is the same as (Finkel and Manning, 2009b), which is also based on the discriminative parser discussed in the previous section. The parse tree structure is augmented with named entity information; see Figure 4 for an example. The features in the joint model are designed in a manner that fits well with the hierarchical joint model: some are over just the parse structure, some are over just the named entities, and some are over the joint structure. The joint model shares the NER and parse features with the respective single-task models. Features over the joint structure only appear in the joint model, and their weights are only indirectly influenced by the singly-annotated data.

In the parsing model, the grammar consists of only the rules observed in the training data. In the joint model, the grammar is augmented with ad-

	Training		Testing	
	Range	# Sent.	Range	# Sent.
ABC	0–55	1195	56–69	199
MNB	0–17	509	18–25	245
NBC	0–29	589	30–39	149
PRI	0–89	1704	90–112	394
VOA	0–198	1508	199–264	385

Table 1: Training and test set sizes for the five datasets in sentences. The file ranges refer to the numbers within the names of the original OntoNotes files.

ditional joint rules which are composed by adding named entity information to existing parse rules. Because the grammars are based on the observed data, and the two models have different data, they will have somewhat different grammars. In our hierarchical joint model, we added all observed rules from the joint data (stripped of named entity information) to the parse-only grammar, and we added all observed rules from the parse-only data to the grammar for the joint model, and augmented them with named entity information in the same manner as the rules observed in the joint data.

Earlier we said that the NER-only model uses identical named entity features as the joint model (and similarly for the parse-only model), but this is not quite true. They use identical *feature templates*, such as *word*, but different realizations of those features will occur with the different datasets. For instance, the NER-only model may have *word=Nigel* as a feature, but because *Nigel* never occurs in the joint data, that feature is never manifested and no weight is learned for it. We deal with this similarly to how we dealt with the grammar: if a named entity feature occurs in either the joint data or the NER-only data, then both models will learn a weight for that feature. We do the same thing for the parse features. This modeling decision gives the joint model access to potentially useful features to which it would not have had access if it were not part of the hierarchical model.[5]

5 Experiments and Discussion

We compared our hierarchical joint model to a regular (non-hierarchical) joint model, and to parse-only and NER-only models. Our baseline experiments were modeled after those in (Finkel and Manning, 2009b), and while our results were not identical (we updated to a newer release of the data), we had similar results and found the same general trends with respect to how the joint model improved on the single models. We used OntoNotes 3.0 (Hovy et al., 2006), and made the same data modifications as (Finkel and Manning, 2009b) to ensure consistency between the parsing and named entity annotations. Table 2 has our complete set of results, and Table 1 gives the number of training and test sentences. For each section of the data (ABC, MNB, NBC, PRI, VOA) we ran experiments training a linear-chain CRF on only the named entity information, a CRF-CFG parser on only the parse information, a joint parser and named entity recognizer, and our hierarchical model. For the hierarchical model, we used the CNN portion of the data (5093 sentences) for the extra named entity data (and ignored the parse trees) and the remaining portions combined for the extra parse data (and ignored the named entity annotations). We used $\sigma_* = 1.0$ and $\sigma_m = 0.1$, which were chosen based on early experiments on development data. Small changes to σ_m do not appear to have much influence, but larger changes do. We similarly decided how many iterations to run stochastic gradient descent for (20) based on early development data experiments. We did not run this experiment on the CNN portion of the data, because the CNN data was already being used as the extra NER data.

As Table 2 shows, the hierarchical model did substantially better than the joint model overall, which is not surprising given the extra data to which it had access. Looking at the smaller corpora (NBC and MNB) we see the largest gains, with both parse and NER performance improving by about 8% F1. ABC saw about a 6% gain on both tasks, and VOA saw a 1% gain on both. Our one negative result is in the PRI portion: parsing improves slightly, but NER performance decreases by almost 2%. The same experiment on development data resulted in a performance increase, so we are not sure why we saw a decrease here. One general trend, which is not surprising, is that the hierarchical model helps the smaller datasets more than the large ones. The source of this is twofold: lower baselines are generally easier to improve upon, and the larger corpora had less singly-annotated data to provide improvements, because it was composed of the remaining, smaller, sections of OntoNotes. We found it interesting that the gains tended to be similar on both tasks for all datasets, and believe this fact is due to our use of roughly the same amount of singly-annotated data for both parsing and NER.

One possible conflating factor in these experiments is that of domain drift. While we tried to

[5] In the non-hierarchical setting, you could include those features in the optimization, but, because there would be no evidence about them, their weights would be zero due to regularization.

		Parse Labeled Bracketing			Named Entities		
		Precision	Recall	F_1	Precision	Recall	F_1
ABC	Just Parse	69.8%	69.9%	69.8%		–	
	Just NER		–		77.0%	75.1%	76.0%
	Baseline Joint	70.2%	70.5%	70.3%	79.2%	76.5%	77.8%
	Hierarchical Joint	75.5%	74.4%	**74.9%**	85.1%	82.7%	**83.9%**
MNB	Just Parse	61.7%	65.5%	63.6%		–	
	Just NER		–		69.6%	49.0%	57.5%
	Baseline Joint	61.7%	66.2%	63.9%	70.9%	63.5%	67.0%
	Hierarchical Joint	72.6%	70.2%	**71.4%**	74.4%	75.5%	**74.9%**
NBC	Just Parse	59.9%	63.9%	61.8%		–	
	Just NER		–		63.9%	60.9%	62.4%
	Baseline Joint	59.3%	64.2%	61.6%	68.9%	62.8%	65.7%
	Hierarchical Joint	70.4%	69.9%	**70.2%**	72.9%	74.0%	**73.4%**
PRI	Just Parse	78.6%	77.0%	76.9%		–	
	Just NER		–		81.3%	77.8%	79.5%
	Baseline Joint	78.0%	78.6%	78.3%	86.3%	86.0%	**86.2%**
	Hierarchical Joint	79.2%	78.5%	**78.8%**	84.2%	85.5%	84.8%
VOA	Just Parse	77.5%	76.5%	77.0%		–	
	Just NER		–		85.2%	80.3%	82.7%
	Baseline Joint	77.2%	77.8%	77.5%	87.5%	86.7%	87.1%
	Hierarchical Joint	79.8%	77.8%	**78.8%**	87.7%	88.9%	**88.3%**

Table 2: Full parse and NER results for the six datasets. Parse trees were evaluated using evalB, and named entities were scored using micro-averaged F-measure (conlleval).

get the most similar annotated data available – data which was annotated by the same annotators, and all of which is broadcast news – these are still different domains. While this is likely to have a negative effect on results, we also believe this scenario to be a more realistic than if it were to also be data drawn from the exact same distribution.

6 Conclusion

In this paper we presented a novel method for improving joint modeling using additional data which has not been labeled with the entire joint structure. While conventional wisdom says that adding more training data should always improve performance, this work is the first to our knowledge to incorporate singly-annotated data into a joint model, thereby providing a method for this additional data, which cannot be directly used by the non-hierarchical joint model, to help improve joint modeling performance. We built single-task models for the non-jointly labeled data, designing those single-task models so that they have features in common with the joint model, and then linked all of the different single-task and joint models via a hierarchical prior. We performed experiments on joint parsing and named entity recognition, and found that our hierarchical joint model substantially outperformed a joint model which was trained on only the jointly annotated data.

Future directions for this work include automatically learning the variances, σ_m and σ_* in the hierarchical model, so that the degree of information sharing between the models is optimized based on the training data available. We are also interested in ways to modify the objective function to place more emphasis on learning a good joint model, instead of equally weighting the learning of the joint and single-task models.

Acknowledgments

Many thanks to Daphne Koller for discussions which led to this work, and to Richard Socher for his assistance and input. Thanks also to our anonymous reviewers and Yoav Goldberg for useful feedback on an earlier draft of this paper.

This material is based upon work supported by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the Air Force Research Laboratory (AFRL). The first author is additionally supported by a Stanford Graduate Fellowship.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 665–672, Morristown, NJ, USA. Association for Computational Linguistics.

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 1–9, Morristown, NJ, USA. Association for Computational Linguistics.

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*.

J. Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. In *Machine Learning*, volume 28.

R. Caruana. 1997. Multitask learning. In *Machine Learning*, volume 28.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 1997*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Gal Elidan, Benjamin Packer, Geremy Heitz, and Daphne Koller. 2008. Convex point estimation using undirected bayesian transfer hierarchies. In *UAI 2008*.

T. Evgeniou, C. Micchelli, and M. Pontil. 2005. Learning multiple tasks with kernel methods. In *Journal of Machine Learning Research*.

Jenny Rose Finkel and Christopher D. Manning. 2009a. Hierarchical bayesian domain adaptation. In *Proceedings of the North American Association of Computational Linguistics (NAACL 2009)*.

Jenny Rose Finkel and Christopher D. Manning. 2009b. Joint parsing and named entity recognition. In *Proceedings of the North American Association of Computational Linguistics (NAACL 2009)*.

Jenny Rose Finkel and Christopher D. Manning. 2009c. Nested named entity recognition. In *Proceedings of EMNLP 2009*.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based conditional random field parsing. In *ACL/HLT-2008*.

Andrew Gelman and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

A. Gelman, J. B. Carlin, H. S. Stern, and Donald D. B. Rubin. 2003. *Bayesian Data Analysis*. Chapman & Hall.

Yoav Goldberg and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL-08: HLT*, pages 371–379, Columbus, Ohio, June. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *HLT-NAACL 2006*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *CoNLL '08: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187, Morristown, NJ, USA. Association for Computational Linguistics.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *In 6th Applied Natural Language Processing Conference*, pages 226–233.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, Manchester, UK.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Conference on Natural Language Learning (CoNLL)*.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 486–494, Suntec, Singapore, August. Association for Computational Linguistics.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, 8.

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning gaussian processes from multiple tasks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL 2008*.

Detecting Errors in Automatically-Parsed Dependency Relations

Markus Dickinson
Indiana University
md7@indiana.edu

Abstract

We outline different methods to detect errors in automatically-parsed dependency corpora, by comparing so-called dependency rules to their representation in the training data and flagging anomalous ones. By comparing each new rule to every relevant rule from training, we can identify parts of parse trees which are likely erroneous. Even the relatively simple methods of comparison we propose show promise for speeding up the annotation process.

1 Introduction and Motivation

Given the need for high-quality dependency parses in applications such as statistical machine translation (Xu et al., 2009), natural language generation (Wan et al., 2009), and text summarization evaluation (Owczarzak, 2009), there is a corresponding need for high-quality dependency annotation, for the training and evaluation of dependency parsers (Buchholz and Marsi, 2006). Furthermore, parsing accuracy degrades unless sufficient amounts of labeled training data from the same domain are available (e.g., Gildea, 2001; Sekine, 1997), and thus we need larger and more varied annotated treebanks, covering a wide range of domains. However, there is a bottleneck in obtaining annotation, due to the need for manual intervention in annotating a treebank. One approach is to develop automatically-parsed corpora (van Noord and Bouma, 2009), but a natural disadvantage with such data is that it contains parsing errors. Identifying the most problematic parses for human post-processing could combine the benefits of automatic and manual annotation, by allowing a human annotator to efficiently correct automatic errors. We thus set out in this paper to detect errors in automatically-parsed data.

If annotated corpora are to grow in scale and retain a high quality, annotation errors which arise from automatic processing must be minimized, as errors have a negative impact on training and evaluation of NLP technology (see discussion and references in Boyd et al., 2008, sec. 1). There is work on detecting errors in dependency corpus annotation (Boyd et al., 2008), but this is based on finding inconsistencies in annotation for identical recurring strings. This emphasis on identical strings can result in high precision, but many strings do not recur, negatively impacting the recall of error detection. Furthermore, since the same strings often receive the same automatic parse, the types of inconsistencies detected are likely to have resulted from manual annotation. While we can build from the insight that simple methods can provide reliable annotation checks, we need an approach which relies on more general properties of the dependency structures, in order to develop techniques which work for automatically-parsed corpora.

Developing techniques to detect errors in parses in a way which is independent of corpus and parser has fairly broad implications. By using only the information available in a training corpus, the methods we explore are applicable to annotation error detection for either hand-annotated or automatically-parsed corpora and can also provide insights for parse reranking (e.g., Hall and Novák, 2005) or parse revision (Attardi and Ciaramita, 2007). Although we focus only on detecting errors in automatically-parsed data, similar techniques have been applied for hand-annotated data (Dickinson, 2008; Dickinson and Foster, 2009).

Our general approach is based on extracting a grammar from an annotated corpus and comparing *dependency rules* in a new (automatically-annotated) corpus to the grammar. Roughly speaking, if a dependency rule—which represents all the dependents of a head together (see section 3.1)—does not fit well with the grammar, it is flagged as potentially erroneous. The methods do not have to be retrained for a given parser's output (e.g.,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 729–738,
Uppsala, Sweden, 11-16 July 2010.

Campbell and Johnson, 2002), but work by comparing any tree to what is in the training grammar (cf. also approaches stacking hand-written rules on top of other parsers (Bick, 2007)).

We propose to flag erroneous parse rules, using information which reflects different grammatical properties: POS lookup, bigram information, and full rule comparisons. We build on a method to detect so-called ad hoc rules, as described in section 2, and then turn to the main approaches in section 3. After a discussion of a simple way to flag POS anomalies in section 4, we evaluate the different methods in section 5, using the outputs from two different parsers. The methodology proposed in this paper is easy to implement and independent of corpus, language, or parser.

2 Approach

We take as a starting point two methods for detecting *ad hoc* rules in constituency annotation (Dickinson, 2008). Ad hoc rules are CFG productions extracted from a treebank which are "used for specific constructions and unlikely to be used again," indicating annotation errors and rules for ungrammaticalities (see also Dickinson and Foster, 2009).

Each method compares a given CFG rule to all the rules in a treebank grammar. Based on the number of similar rules, a score is assigned, and rules with the lowest scores are flagged as potentially ad hoc. This procedure is applicable whether the rules in question are from a new data set—as in this paper, where parses are compared to a training data grammar—or drawn from the treebank grammar itself (i.e., an internal consistency check).

The two methods differ in how the comparisons are done. First, the *bigram method* abstracts a rule to its bigrams. Thus, a rule such as NP → JJ NN provides support for NP → DT JJ JJ NN, in that it shares the JJ NN sequence. By contrast, in the other method, which we call the *whole rule method*,[1] a rule is compared in its totality to the grammar rules, using Levenshtein distance. There is no abstraction, meaning all elements are present—e.g., NP → DT JJ JJ NN is very similar to NP → DT JJ NN because the sequences differ by only one category.

While previously used for constituencies, what is at issue is simply the *valency* of a rule, where by valency we refer to a head and its entire set of arguments and adjuncts (cf. Przepiórkowski, 2006)—that is, a head and all its dependents. The methods work because we expect there to be regularities in valency structure in a treebank grammar; non-conformity to such regularities indicates a potential problem.

3 Ad hoc rule detection

3.1 An appropriate representation

To capture valency, consider the dependency tree from the Talbanken05 corpus (Nilsson and Hall, 2005) in figure 1, for the Swedish sentence in (1), which has four dependency pairs.[2]

(1) Det går bara inte ihop .
it goes just not together
'It just doesn't add up.'

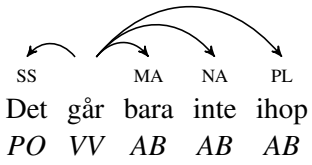


Figure 1: Dependency graph example

On a par with constituency rules, we define a grammar *rule* as a dependency relation rewriting as a head with its sequence of POS/dependent pairs (cf. Kuhlmann and Satta, 2009), as in figure 2. This representation supports the detection of idiosyncracies in valency.[3]

1. TOP → root ROOT:VV
2. ROOT → SS:PO VV MA:AB NA:AB PL:AB
3. SS → PO
4. MA → AB
5. NA → AB
6. PL → AB

Figure 2: Rule representation for (1)

For example, for the ROOT category, the head is a verb (VV), and it has 4 dependents. The extent to which this rule is odd depends upon whether comparable rules—i.e., other ROOT rules or other VV rules (see section 3.2)—have a similar set of dependents. While many of the other rules seem rather spare, they provide useful information, showing categories which have no dependents. With a TOP rule, we have a rule for every

[1] This is referred to *whole daughters* in Dickinson (2008), but the meaning of "daughters" is less clear for dependencies.

[2] Category definitions are in appendix A.

[3] Valency is difficult to define for coordination and is specific to an annotation scheme. We leave this for the future.

head, including the virtual root. Thus, we can find anomalous rules such as TOP → root ROOT:AV ROOT:NN, where multiple categories have been parsed as ROOT.

3.2 Making appropriate comparisons

In comparing rules, we are trying to find evidence that a particular (parsed) rule is valid by examining the evidence from the (training) grammar.

Units of comparison To determine similarity, one can compare dependency relations, POS tags, or both. Valency refers to both properties, e.g., verbs which allow verbal (POS) subjects (dependency). Thus, we use the pairs of dependency relations and POS tags as the units of comparison.

Flagging individual elements Previous work scored only entire rules, but some dependencies are problematic and others are not. Thus, our methods score individual elements of a rule.

Comparable rules We do not want to compare a rule to all grammar rules, only to those which *should* have the same valents. Comparability could be defined in terms of a rule's dependency relation (LHS) or in terms of its head. Consider the four different object (OO) rules in (2). These vary a great deal, and much of the variability comes from the fact that they are headed by different POS categories, which tend to have different selectional properties. The head POS thus seems to be predictive of a rule's valency.

(2) a. OO → **PO**
b. OO → DT:EN AT:AJ **NN** ET:VV
c. OO → SS:PO **QV** VG:VV
d. OO → DT:PO AT:AJ **VN**

But we might lose information by ignoring rules with the same left-hand side (LHS). Our approach is thus to take the greater value of scores when comparing to rules *either* with the same dependency relation or with the same head. A rule has multiple chances to prove its value, and low scores will only be for rules without any type of support.

Taking these points together, for a given rule of interest r, we assign a score (S) to each element e_i in r, where $r = e_1...e_m$ by taking the maximum of scores for rules with the same head (h) or same LHS (lhs), as in (3). For the first element in (2b), for example, S(DT:EN) = max{s(DT:EN, NN), s(DT:EN, OO)}. The question is now how we define $s(e_i, c)$ for the comparable element c.

(3) $S(e_i) = \max\{s(e_i, h), s(e_i, lhs)\}$

3.3 Whole rule anomalies

3.3.1 Motivation

The whole rule method compares a list of a rule's dependents to rules in a database, and then flags rule elements without much support. By using all dependents as a basis for comparison, this method detects improper dependencies (e.g., an adverb modifying a noun), dependencies in the wrong overall location of a rule (e.g., an adverb before an object), and rules with unnecessarily long argument structures. For example, in (4), we have an improper relation between *skall* ('shall') and *sambeskattas* ('be taxed together'), as in figure 3. It is parsed as an adverb (AA), whereas it should be a verb group (VG). The rule for this part of the tree is +F → ++:++ SV AA:VV, and the AA:VV position will be low-scoring because the ++:++ SV context does not support it.

(4) Makars övriga inkomster är B-inkomster
spouses' other incomes are B-incomes
och *skall* som tidigare *sambeskattas* .
and shall as previously be taxed togeher .
'The other incomes of spouses are B-incomes and shall, as previously, be taxed together.'

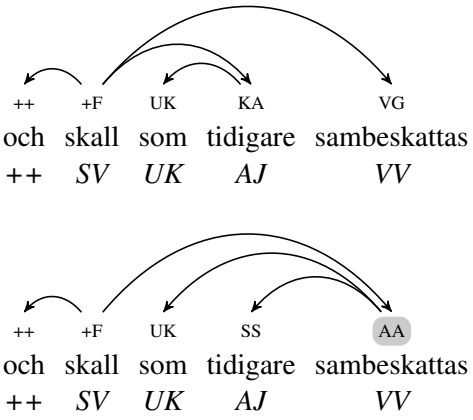


Figure 3: Wrong label (top=gold, bottom=parsed)

3.3.2 Implementation

The method we use to determine similarity arises from considering what a rule is like without a problematic element. Consider +F → ++:++ SV AA:VV from figure 3, where AA should be a different category (VG). The rule without this error, +F → ++:++ SV, starts several rules in the

training data, including some with VG:VV as the next item. The subrule ++:++ SV seems to be reliable, whereas the subrules containing AA:VV (++:++ AA:VV and SV AA:VV) are less reliable. We thus determine reliability by seeing how often each subsequence occurs in the training rule set.

Throughout this paper, we use the term *subrule* to refer to a rule subsequence which is exactly one element shorter than the rule it is a component of. We examine subrules, counting their frequency *as subrules*, not as complete rules. For example, TOP rules with more than one dependent are problematic, e.g., TOP → root ROOT:AV ROOT:NN. Correspondingly, there are no rules with *three* elements containing the subrule root ROOT:AV.

We formalize this by setting the score $s(e_i, c)$ equal to the summation of the frequencies of all comparable subrules containing e_i from the training data, as in (5), where B is the set of subrules of r with length one less.

(5) $s(e_i, c) = \sum_{sub \in B : e_i \in sub} C(sub, c)$

For example, with c = +F, the frequency of +F → ++:++ SV as a subrule is added to the scores for ++:++ and SV. In this case, +F → ++:++ SV **VG:BV**, +F → ++:++ SV **VG:AV**, and +F → ++:++ SV **VG:VV** all add support for +F → ++:++ SV being a legitimate subrule. Thus, ++:++ and SV are less likely to be the sources of any problems. Since +F → SV AA:VV and +F → ++:++ AA:VV have very little support in the training data, AA:VV receives a low score.

Note that the subrule count $C(sub, c)$ is different than counting the number of rules containing a subrule, as can be seen with identical elements. For example, for SS → VN ET:PR ET:PR, C(VN ET:PR, SS) = 2, in keeping with the fact that there are 2 pieces of evidence for its legitimacy.

3.4 Bigram anomalies

3.4.1 Motivation

The bigram method examines relationships between adjacent sisters, complementing the whole rule method by focusing on local properties. For (6), for example, we find the gold and parsed trees in figure 4. For the long parsed rule TA → PR HD:ID HD:ID **IR:IR AN:RO JR:IR**, all elements get low whole rule scores, i.e., are flagged as potentially erroneous. But only the final elements have anomalous bigrams: HD:ID IR:IR, IR:IR AN:RO, and AN:RO JR:IR all never occur.

(6) När det gäller inkomståret 1971 (
when it concerns the income year 1971 (
taxeringsåret 1972) skall barnet ...
assessment year 1972) shall the child ...
'Concerning the income year of 1971 (assessment year 1972), the child ...'

3.4.2 Implementation

To obtain a bigram score for an element, we simply add together the bigrams which contain the element in question, as in (7).

(7) $s(e_i, c) = C(e_{i-1}\mathbf{e_i}, c) + C(\mathbf{e_i}e_{i+1}, c)$

Consider the rule from figure 4. With $c = TA$, the bigram HD:ID IR:IR never occurs, so both HD:ID and IR:IR get 0 added to their score. HD:ID **HD:ID**, however, is a frequent bigram, so it adds weight to HD:ID, i.e., positive evidence comes from the bigram on the left. If we look at IR:IR, on the other hand, **IR:IR** AN:RO occurs 0 times, and so IR:IR gets a total score of 0.

Both scoring methods treat each element independently. Every single element could be given a low score, even though once one is corrected, another would have a higher score. Future work can examine factoring in all elements at once.

4 Additional information

The methods presented so far have limited definitions of comparability. As using complementary information has been useful in, e.g., POS error detection (Loftsson, 2009), we explore other simple comparable properties of a dependency grammar. Namely, we include: a) frequency information of an overall dependency rule and b) information on how likely each dependent is to be in a relation with its head, described next.

4.1 Including POS information

Consider PA → SS:NN XX:XX HV OO:VN, as illustrated in figure 5 for the sentence in (8). This rule is entirely correct, yet the XX:XX position has low whole rule and bigram scores.

(8) Uppgift om vilka orter som
information of which neighborhood who
har utkörning finner Ni också i ...
has delivery find you also in ...
'You can also find information about which neighborhoods have delivery services in ...'

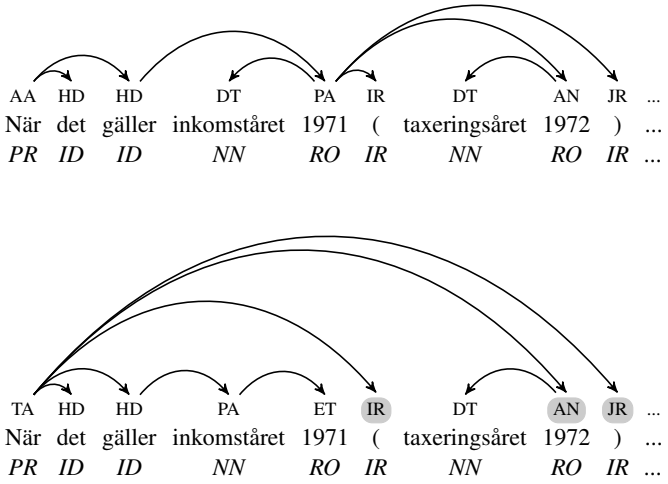


Figure 4: A rule with extra dependents (top=gold, bottom=parsed)

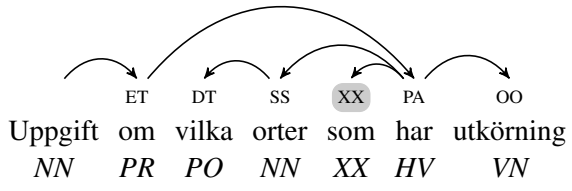


Figure 5: Overflagging (gold=parsed)

One method which does not have this problem of overflagging uses a "lexicon" of POS tag pairs, examining relations between POS, irrespective of position. We extract POS pairs, note their dependency relation, and add a L/R to the label to indicate which is the head (Boyd et al., 2008). Additionally, we note how often two POS categories occur as a non-depenency, using the label NIL, to help determine whether there should be any attachment. We generate NILs by enumerating all POS pairs in a sentence. For example, from figure 5, the parsed POS pairs include NN PR $\mapsto$ ET-L, NN PO $\mapsto$ NIL, etc.

We convert the frequencies to probabilities. For example, of 4 total occurrences of XX HV in the training data, 2 are XX-R (cf. figure 5). A probability of 0.5 is quite high, given that NILs are often the most frequent label for POS pairs.

5 Evaluation

In evaluating the methods, our main question is: how accurate are the dependencies, in terms of both attachment and labeling? We therefore currently examine the scores for elements functioning as dependents in a rule. In figure 5, for example, for *har* ('has'), we look at its score within ET $\rightarrow$ PR **PA:HV** and not when it functions as a head, as in PA $\rightarrow$ SS:NN XX:XX **HV** OO:VN.

Relatedly, for each method, we are interested in whether elements with scores below a threshold have worse attachment accuracy than scores above, as we predict they do. We can measure this by scoring each testing data position below the threshold as a 1 if it has the correct head and dependency relation and a 0 otherwise. These are simply labeled attachment scores (LAS). Scoring separately for positions above and below a threshold views the task as one of sorting parser output into two bins, those more or less likely to be correctly parsed. For development, we also report unlabeled attachement scores (UAS).

Since the goal is to speed up the post-editing of corpus data by flagging erroneous rules, we also report the precision and recall for error detection. We count either attachment or labeling errors as an error, and precision and recall are measured with respect to how many errors are found below the threshold. For development, we use two F-scores to provide a measure of the settings to examine across language, corpus, and parser conditions: the balanced F_1 measure and the $F_{0.5}$ measure, weighing precision twice as much. Precision is likely more important in this context, so as to prevent annotators from sorting through too many false positives. In practice, one way to use these methods is to start with the lowest thresholds and work upwards until there are too many non-errors.

To establish a basis for comparison, we compare

method performance to a parser on its own.[4] By examining the parser output without any automatic assistance, how often does a correction need to be made?

5.1 The data

All our data comes from the CoNLL-X Shared Task (Buchholz and Marsi, 2006), specifically the 4 data sets freely available online. We use the Swedish Talbanken data (Nilsson and Hall, 2005) and the transition-based dependency parser MaltParser (Nivre et al., 2007), with the default settings, for developing the method. To test across languages and corpora, we use MaltParser on the other 3 corpora: the Danish DDT (Kromann, 2003), Dutch Alpino (van der Beek et al., 2002), and Portuguese Bosque data (Afonso et al., 2002). Then, we present results using the graph-based parser MSTParser (McDonald and Pereira, 2006), again with default settings, to test the methods across parsers. We use the gold standard POS tags for all experiments.

5.2 Development data

In the first line of table 1, we report the baseline MaltParser accuracies on the Swedish test data, including baseline error detection precision (=1-LAS_b), recall, and (the best) F-scores. In the rest of table 1, we report the best-performing results for each of the methods,[5] providing the number of rules below and above a particular threshold, along with corresponding UAS and LAS values. To get the raw number of identified rules, multiply the number of corpus position below a threshold (b) times the error detection precision (P). For example, the bigram method with a threshold of 39 leads to finding 283 errors ($455 \times .622$).

Dependency elements with frequency below the lowest threshold have lower attachment scores (66.6% vs. 90.1% LAS), showing that simply using a complete rule helps sort dependencies. However, frequency thresholds have fairly low precision, i.e., 33.4% at their best. The whole rule and bigram methods reveal greater precision in identifying problematic dependencies, isolating elements with lower UAS and LAS scores than with frequency, along with corresponding greater precision and F-scores. The bigram method is more fine-grained, identifying small numbers of rule elements at each threshold, resulting in high error detection precision. With a threshold of 39, for example, we find over a quarter of the parser errors with 62% precision, from this one piece of information. For POS information, we flag 23.6% of the cases with over 60% precision (at 81.6).

Taking all these results together, we can begin to sort more reliable from less reliable dependency tree elements, using very simple information. Additionally, these methods naturally group cases together by linguistic properties (e.g., adverbial-verb dependencies within a particualr context), allowing a human to uncover the principle behind parse failure and ajudicate similar cases at the same time (cf. Wallis, 2003).

5.3 Discussion

Examining some of the output from the Talbanken test data by hand, we find that a prominent cause of false positives, i.e., correctly-parsed cases with low scores, stems from low-frequency dependency-POS label pairs. If the dependency rarely occurs in the training data with the particular POS, then it receives a low score, regardless of its context. For example, the parsed rule TA $\rightarrow$ **IG:IG** RO has a correct dependency relation (IG) between the POS tags IG and its head RO, yet is assigned a whole rule score of 2 and a bigram score of 20. It turns out that IG:IG only occurs 144 times in the training data, and in 11 of those cases (7.6%) it appears immediately before RO. One might consider normalizing the scores based on overall frequency or adjusting the scores to account for other dependency rules in the sentence: in this case, there may be no better attachment.

Other false positives are correctly-parsed elements that are a part of erroneous rules. For instance, in AA $\rightarrow$ UK:UK SS:PO TA:AJ AV SP:AJ OA:PR **+F:HV** +F:HV, the first +F:HV is correct, yet given a low score (0 whole rule, 1 bigram). The following and erroneous +F:HV is similarly given a low score. As above, such cases might be handled by looking for attachments in other rules (cf. Attardi and Ciaramita, 2007), but these cases should be relatively unproblematic for hand-correction, given the neighboring error.

We also examined false negatives, i.e., errors with high scores. There are many examples of PR PA:NN rules, for instance, with the NN improp-

[4] One may also use parser confidence or parser revision methods as a basis of comparison, but we are aware of no systematic evaluation of these approaches for detecting errors.

[5] Freq=rule frequency, WR=whole rule, Bi=bigram, POS=POS-based (POS scores multiplied by 10,000)

Score	Thr.	b	a	UAS_b	LAS_b	UAS_a	LAS_a	P	R	F_1	$F_{0.5}$
None	n/a	5656	0	87.4%	82.0%	0%	0%	18.0%	100%	30.5%	21.5%
Freq	0	1951	3705	76.6%	66.6%	93.1%	90.1%	33.4%	64.1%	**43.9%**	**36.9%**
WR	0	894	4762	64.7%	54.0%	91.7%	87.3%	46.0%	40.5%	43.0%	**44.8%**
	6	1478	4178	71.1%	60.9%	93.2%	89.5%	39.1%	56.9%	**46.4%**	41.7%
Bi	0	56	5600	10.7%	7.1%	88.2%	82.8%	92.9%	5.1%	9.7%	21.0%
	39	455	5201	51.6%	37.8%	90.6%	85.9%	62.2%	27.9%	38.5%	**49.9%**
	431	1685	3971	74.1%	63.7%	93.1%	89.8%	36.3%	60.1%	**45.2%**	39.4%
POS	0	54	5602	27.8%	22.2%	87.4%	82.6%	77.8%	4.1%	7.9%	17.0%
	81.6	388	5268	48.5%	38.4%	90.3%	85.3%	61.6%	23.5%	34.0%	**46.5%**
	763	1863	3793	75.4%	65.8%	93.3%	90.0%	34.2%	62.8%	**44.3%**	37.7%

Table 1: MaltParser results for Talbanken, for select values (b = below, a = above threshold (Thr.))

erly attached, but there are also many correct instances of PR PA:NN. To sort out the errors, one needs to look at lexical knowledge and/or other dependencies in the tree. With so little context, frequent rules with only one dependent are not prime candidates for our methods of error detection.

5.4 Other corpora

We now turn to the parsed data from three other corpora. The Alpino and Bosque corpora are approximately the same size as Talbanken, so we use the same thresholds for them. The DDT data is approximately half the size; to adjust, we simply halve the scores. In tables 2, 3, and 4, we present the results, using the best $F_{0.5}$ and F_1 settings from development. At a glance, we observe that the best method differs for each corpus and depending on an emphasis of precision or recall, with the bigram method generally having high precision.

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5585	73.8%	0%	26.2%	100%
Freq	0	1174	43.2%	81.9%	56.8%	45.6%
WR	0	483	32.5%	77.7%	67.5%	22.3%
	6	787	39.4%	79.4%	60.6%	32.6%
Bi	39	253	33.6%	75.7%	66.4%	11.5%
	431	845	45.6%	78.8%	54.4%	31.4%
POS	81.6	317	51.7%	75.1%	48.3%	10.5%
	763	1767	53.5%	83.2%	46.5%	56.1%

Table 2: MaltParser results for Alpino

For Alpino, error detection is better with frequency than, for example, bigram scores. This is likely due to the fact that Alpino has the smallest label set of any of the corpora, with only 24 dependency labels and 12 POS tags (cf. 64 and 41 in Talbanken, respectively). With a smaller label set, there are less possible bigrams that could be anomalous, but more reliable statistics about a whole rule. Likewise, with fewer possible POS tag pairs, Alpino has lower precision for the low-threshold POS scores than the other corpora.

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5867	82.2%	0%	17.8%	100%
Freq	0	1561	61.2%	89.9%	38.8%	58.1%
WR	0	693	48.1%	86.8%	51.9%	34.5%
	6	1074	54.4%	88.5%	45.6%	47.0%
Bi	39	227	15.4%	84.9%	84.6%	18.4%
	431	776	51.0%	87.0%	49.0%	36.5%
POS	81.6	369	33.3%	85.5%	66.7%	23.6%
	763	1681	60.1%	91.1%	39.9%	64.3%

Table 3: MaltParser results for Bosque

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5852	81.0%	0%	19.0%	100%
Freq	0	1835	65.9%	88.0%	34.1%	56.4%
WR	0	739	53.9%	85.0%	46.1%	30.7%
	3	1109	60.1%	85.9%	39.9%	39.9%
Bi	19.5	185	25.4%	82.9%	74.6%	12.4%
	215.5	884	56.8%	85.4%	43.2%	34.4%
POS	40.8	179	30.2%	82.7%	69.8%	11.3%
	381.5	1214	62.5%	85.9%	37.5%	41.0%

Table 4: MaltParser results for DDT

For the whole rule scores, the DDT data is worse (compare its 46.1% precision with Bosque's 45.6%, with vastly different recall values), which could be due to the smaller training data. One might also consider the qualitative differences in the dependency inventory of DDT compared to the others—e.g., appositions, distinctions in names, and more types of modifiers.

5.5 MSTParser

Turning to the results of running the methods on the output of MSTParser, we find similar but slightly worse values for the whole rule and bigram methods, as shown in tables 5-8. What is

most striking are the differences in the POS-based method for Bosque and DDT (tables 7 and 8), where a large percentage of the test corpus is underneath the threshold. MSTParser is apparently positing fewer distinct head-dependent pairs, as most of them fall under the given thresholds. With the exception of the POS-based method for DDT (where LAS_b is actually higher than LAS_a) the different methods seem to be accurate enough to be used as part of corpus post-editing.

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5656	81.1%	0%	18.9%	100%
Freq	0	3659	65.2%	89.7%	34.8%	64.9%
WR	0	4740	55.7%	86.0%	44.3%	37.9%
	6	4217	59.9%	88.3%	40.1%	53.9%
Bi	39	5183	38.9%	84.9%	61.1%	27.0%
	431	3997	63.2%	88.5%	36.8%	57.1%
POS	81.6	327	42.8%	83.4%	57.2%	17.5%
	763	1764	68.0%	87.0%	32.0%	52.7%

Table 5: MSTParser results for Talbanken

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5585	75.4%	0%	24.6%	100%
Freq	0	1371	49.5%	83.9%	50.5%	50.5%
WR	0	453	40.0%	78.5%	60.0%	19.8%
	6	685	45.4%	79.6%	54.6%	27.2%
Bi	39	226	39.8%	76.9%	60.2%	9.9%
	431	745	48.2%	79.6%	51.8%	28.1%
POS	81.6	570	60.4%	77.1%	39.6%	16.5%
	763	1860	61.9%	82.1%	38.1%	51.6%

Table 6: MSTParser results for Alpino

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5867	82.5%	0%	17.5%	100%
Freq	0	1562	63.9%	89.3%	36.1%	55.0%
WR	0	540	50.6%	85.8%	49.4%	26.0%
	6	985	58.0%	87.5%	42.0%	40.4%
Bi	39	117	34.2%	83.5%	65.8%	7.5%
	431	736	56.4%	86.3%	43.6%	31.3%
POS	81.6	2978	75.8%	89.4%	24.2%	70.3%
	763	3618	74.3%	95.8%	25.7%	90.7%

Table 7: MSTParser results for Bosque

Score	Thr.	b	LAS_b	LAS_a	P	R
None	n/a	5852	82.9%	0%	17.1%	100%
Freq	0	1864	70.3%	88.8%	29.7%	55.3%
WR	0	624	60.6%	85.6%	39.4%	24.6%
	3	1019	65.4%	86.6%	34.6%	35.3%
Bi	19.5	168	28.6%	84.5%	71.4%	12.0%
	215.5	839	61.6%	86.5%	38.4%	32.2%
POS	40.8	5714	83.0%	79.0%	17.0%	97.1%
	381.5	5757	82.9%	80.0%	17.1%	98.1%

Table 8: MSTParser results for DDT

6 Summary and Outlook

We have proposed different methods for flagging the errors in automatically-parsed corpora, by treating the problem as one of looking for anomalous rules with respect to a treebank grammar. The different methods incorporate differing types and amounts of information, notably comparisons among dependency rules and bigrams within such rules. Using these methods, we demonstrated success in sorting well-formed output from erroneous output across language, corpora, and parsers.

Given that the rule representations and comparison methods use both POS and dependency information, a next step in evaluating and improving the methods is to examine automatically POS-tagged data. Our methods should be able to find POS errors in addition to dependency errors. Furthermore, although we have indicated that differences in accuracy can be linked to differences in the granularity and particular distinctions of the annotation scheme, it is still an open question as to which methods work best for which schemes and for which constructions (e.g., coordination).

Acknowledgments

Thanks to Sandra Kübler and Amber Smith for comments on an earlier draft; Yvonne Samuelsson for help with the Swedish translations; the IU Computational Linguistics discussion group for feedback; and Julia Hockenmaier, Chris Brew, and Rebecca Hwa for discussion on the general topic.

A Some Talbanken05 categories

POS tags	
++	coord. conj.
AB	adverb
AJ	adjective
AV	*vara* (be)
EN	indef. article
HV	*ha(va)* (have)
ID	part of idiom
IG	punctuation
IR	parenthesis
NN	noun
PO	pronoun
PR	preposition
RO	numeral
QV	*kunna* (can)
SV	*skola* (will)
UK	sub. conj.
VN	verbal noun
VV	verb
XX	unclassifiable

Dependencies	
++	coord. conj.
+F	main clause coord.
AA	adverbial
AN	apposition
AT	nomainl pre-modifier
DT	determiner
ET	nominal post-modifier
HD	head
IG	punctuation
IR	parenthesis
JR	second parenthesis
KA	comparative adverbial
MA	attitude adverbial
NA	negation adverbial
OO	object
PA	preposition comp.
PL	verb particle
SS	subject
TA	time adverbial
UK	sub. conj.
VG	verb group
XX	unclassifiable

References

Afonso, Susana, Eckhard Bick, Renato Haber and Diana Santos (2002). Floresta Sintá(c)tica: a treebank for Portuguese. In *Proceedings of LREC 2002*. Las Palmas, pp. 1698–1703.

Attardi, Giuseppe and Massimiliano Ciaramita (2007). Tree Revision Learning for Dependency Parsing. In *Proceedings of NAACL-HLT-07*. Rochester, NY, pp. 388–395.

Bick, Eckhard (2007). Hybrid Ways to Improve Domain Independence in an ML Dependency Parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Prague, Czech Republic, pp. 1119–1123.

Boyd, Adriane, Markus Dickinson and Detmar Meurers (2008). On Detecting Errors in Dependency Treebanks. *Research on Language and Computation* 6(2), 113–137.

Buchholz, Sabine and Erwin Marsi (2006). CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNLL-X*. New York City, pp. 149–164.

Campbell, David and Stephen Johnson (2002). A transformational-based learner for dependency grammars in discharge summaries. In *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain*. Phildadelphia, pp. 37–44.

Dickinson, Markus (2008). Ad Hoc Treebank Structures. In *Proceedings of ACL-08*. Columbus, OH.

Dickinson, Markus and Jennifer Foster (2009). Similarity Rules! Exploring Methods for Ad-Hoc Rule Detection. In *Proceedings of TLT-7*. Groningen, The Netherlands.

Gildea, Daniel (2001). Corpus Variation and Parser Performance. In *Proceedings of EMNLP-01*. Pittsburgh, PA.

Hall, Keith and Václav Novák (2005). Corrective Modeling for Non-Projective Dependency Parsing. In *Proceedings of IWPT-05*. Vancouver, pp. 42–52.

Kromann, Matthias Trautner (2003). The Danish Dependency Treebank and the underlying linguistic theory. In *Proceedings of TLT-03*.

Kuhlmann, Marco and Giorgio Satta (2009). Treebank Grammar Techniques for Non-Projective Dependency Parsing. In *Proceedings of EACL-09*. Athens, Greece, pp. 478–486.

Loftsson, Hrafn (2009). Correcting a POS-Tagged Corpus Using Three Complementary Methods. In *Proceedings of EACL-09*. Athens, Greece, pp. 523–531.

McDonald, Ryan and Fernando Pereira (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-06*. Trento.

Nilsson, Jens and Johan Hall (2005). *Reconstruction of the Swedish Treebank Talbanken*. MSI report 05067, Växjö University: School of Mathematics and Systems Engineering.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kübler, Svetoslav Marinov and Erwin Marsi (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2), 95–135.

Owczarzak, Karolina (2009). DEPEVAL(summ): Dependency-based Evaluation for Automatic Summaries. In *Proceedings of ACL-AFNLP-09*. Suntec, Singapore, pp. 190–198.

Przepiórkowski, Adam (2006). What to acquire from corpora in automatic valence acquisition. In Violetta Koseska-Toszewa and Roman Roszko (eds.), *Semantyka a konfrontacja jezykowa, tom 3*, Warsaw: Slawistyczny Ośrodek Wydawniczy PAN, pp. 25–41.

Sekine, Satoshi (1997). The Domain Dependence of Parsing. In *Proceedings of ANLP-96*. Washington, DC.

van der Beek, Leonoor, Gosse Bouma, Robert Malouf and Gertjan van Noord (2002). The Alpino Dependency Treebank. In *Proceedings of CLIN 2001*. Rodopi.

van Noord, Gertjan and Gosse Bouma (2009). Parsed Corpora for Linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*. Athens, pp. 33–39.

Wallis, Sean (2003). Completing Parsed Corpora. In Anne Abeillé (ed.), *Treebanks: Building and using syntactically annoted corpora*, Dordrecht: Kluwer Academic Publishers, pp. 61–71.

Wan, Stephen, Mark Dras, Robert Dale and Cécile Paris (2009). Improving Grammaticality in Sta-

tistical Sentence Generation: Introducing a Dependency Spanning Tree Algorithm with an Argument Satisfaction Model. In *Proceedings of EACL-09*. Athens, Greece, pp. 852–860.

Xu, Peng, Jaeho Kang, Michael Ringgaard and Franz Och (2009). Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages. In *Proceedings of NAACL-HLT-09*. Boulder, Colorado, pp. 245–253.

Boosting-based System Combination for Machine Translation

Tong Xiao, Jingbo Zhu, Muhua Zhu, Huizhen Wang

Natural Language Processing Lab.
Northeastern University, China
{xiaotong,zhujingbo,wanghuizhen}@mail.neu.edu.cn
zhumuhua@gmail.com

Abstract

In this paper, we present a simple and effective method to address the issue of how to generate diversified translation systems from a single Statistical Machine Translation (SMT) engine for system combination. Our method is based on the framework of boosting. First, a sequence of *weak* translation systems is generated from a baseline system in an iterative manner. Then, a *strong* translation system is built from the ensemble of these weak translation systems. To adapt boosting to SMT system combination, several key components of the original boosting algorithms are redesigned in this work. We evaluate our method on Chinese-to-English Machine Translation (MT) tasks in three baseline systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based system. The experimental results on three NIST evaluation test sets show that our method leads to significant improvements in translation accuracy over the baseline systems.

1 Introduction

Recent research on Statistical Machine Translation (SMT) has achieved substantial progress. Many SMT frameworks have been developed, including phrase-based SMT (Koehn et al., 2003), hierarchical phrase-based SMT (Chiang, 2005), syntax-based SMT (Eisner, 2003; Ding and Palmer, 2005; Liu et al., 2006; Galley et al., 2006; Cowan et al., 2006), etc. With the emergence of various structurally different SMT systems, more and more studies are focused on combining multiple SMT systems for achieving higher translation accuracy rather than using a single translation system.

The basic idea of system combination is to extract or generate a translation by voting from an ensemble of translation outputs. Depending on how the translation is combined and what voting strategy is adopted, several methods can be used for system combination, e.g. sentence-level combination (Hildebrand and Vogel, 2008) simply selects one from original translations, while some more sophisticated methods, such as word-level and phrase-level combination (Matusov et al., 2006; Rosti et al., 2007), can generate new translations differing from any of the original translations.

One of the key factors in SMT system combination is the diversity in the ensemble of translation outputs (Macherey and Och, 2007). To obtain diversified translation outputs, most of the current system combination methods require multiple translation engines based on different models. However, this requirement cannot be met in many cases, since we do not always have the access to multiple SMT engines due to the high cost of developing and tuning SMT systems. To reduce the burden of system development, it might be a nice way to combine a set of translation systems built from a single translation engine. A key issue here is how to generate an ensemble of diversified translation systems from a single translation engine in a principled way.

Addressing this issue, we propose a boosting-based system combination method to learn a combined translation system from a single SMT engine. In this method, a sequence of *weak* translation systems is generated from a baseline system in an iterative manner. In each iteration, a new weak translation system is learned, focusing more on the sentences that are relatively poorly translated by the previous weak translation system. Finally, a *strong* translation system is built from the ensemble of the weak translation systems.

Our experiments are conducted on Chinese-to-English translation in three state-of-the-art SMT systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 739–748,
Uppsala, Sweden, 11-16 July 2010.

Input: a model u, a sequence of (training) samples $\{(f_1, \mathbf{r}_1), ..., (f_m, \mathbf{r}_m)\}$ where f_i is the i-th source sentence, and $\mathbf{r}_i$ is the set of reference translations for f_i.
Output: a new translation system

Initialize: $D_1(i) = 1 / m$ for all $i = 1, ..., m$
For $t = 1, ..., T$

1. Train a translation system $u(\boldsymbol{\lambda}^*_t)$ on $\{(f_i, \mathbf{r}_i)\}$ using distribution D_t
2. Calculate the error rate ε_t of $u(\boldsymbol{\lambda}^*_t)$ on $\{(f_i, \mathbf{r}_i)\}$
3. Set

$$\alpha_t = \frac{1}{2}\ln(\frac{1+\varepsilon_t}{\varepsilon_t}) \quad (3)$$

4. Update weights

$$D_{t+1}(i) = \frac{D_t(i)e^{\alpha_t \cdot l_i}}{Z_t} \quad (4)$$

where l_i is the loss on the i-th training sample, and Z_t is the normalization factor.

Output the final system:

$$v(u(\boldsymbol{\lambda}^*_1), ..., u(\boldsymbol{\lambda}^*_T))$$

Figure 1: Boosting-based System Combination

system. All the systems are evaluated on three NIST MT evaluation test sets. Experimental results show that our method leads to significant improvements in translation accuracy over the baseline systems.

2 Background

Given a source string f, the goal of SMT is to find a target string e^* by the following equation.

$$e^* = \arg\max_e(\Pr(e \mid f)) \quad (1)$$

where $\Pr(e \mid f)$ is the probability that e is the translation of the given source string f. To model the posterior probability $\Pr(e \mid f)$, most of the state-of-the-art SMT systems utilize the log-linear model proposed by Och and Ney (2002), as follows,

$$\Pr(e \mid f) = \frac{\exp(\sum_{m=1}^{M}\lambda_m \cdot h_m(f,e))}{\sum_{e'}\exp(\sum_{m=1}^{M}\lambda_m \cdot h_m(f,e'))} \quad (2)$$

where $\{h_m(f, e) \mid m = 1, ..., M\}$ is a set of *features*, and λ_m is the *feature weight* corresponding to the m-th feature. $h_m(f, e)$ can be regarded as a function that maps every pair of source string f and target string e into a non-negative value, and λ_m can be viewed as the contribution of $h_m(f, e)$ to the overall score $\Pr(e \mid f)$.

In this paper, u denotes a log-linear model that has M fixed features $\{h_1(f,e), ..., h_M(f,e)\}$, $\boldsymbol{\lambda} = \{\lambda_1, ..., \lambda_M\}$ denotes the M parameters of u, and $u(\boldsymbol{\lambda})$ denotes a SMT system based on u with parameters $\boldsymbol{\lambda}$. Generally, $\boldsymbol{\lambda}$ is trained on a training data set[1] to obtain an optimized weight vector $\boldsymbol{\lambda}^*$ and consequently an optimized system $u(\boldsymbol{\lambda}^*)$.

3 Boosting-based System Combination for Single Translation Engine

Suppose that there are T available SMT systems $\{u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T)\}$, the task of system combination is to build a new translation system $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$ from $\{u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T)\}$. Here $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$ denotes the *combination system* which combines translations from the ensemble of the output of each $u_i(\boldsymbol{\lambda}^*_i)$. We call $u_i(\boldsymbol{\lambda}^*_i)$ a *member system* of $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$. As discussed in Section 1, the diversity among the outputs of member systems is an important factor to the success of system combination. To obtain diversified member systems, traditional methods concentrate more on using structurally different member systems, that is $u_1 \neq u_2 \neq ... \neq u_T$. However, this constraint condition cannot be satisfied when multiple translation engines are not available.

In this paper, we argue that the diversified member systems can also be generated from a single engine $u(\boldsymbol{\lambda}^*)$ by adjusting the weight vector $\boldsymbol{\lambda}^*$ in a principled way. In this work, we assume that $u_1 = u_2 = ... = u_T = u$. Our goal is to find a series of $\boldsymbol{\lambda}^*_i$ and build a combined system from $\{u(\boldsymbol{\lambda}^*_i)\}$. To achieve this goal, we propose a

[1] The data set used for weight training is generally called *development set* or *tuning set* in the SMT field. In this paper, we use the term *training set* to emphasize the training of log-linear model.

boosting-based system combination method (Figure 1).

Like other boosting algorithms, such as AdaBoost (Freund and Schapire, 1997; Schapire, 2001), the basic idea of this method is to use weak systems (*member systems*) to form a strong system (*combined system*) by repeatedly calling weak system trainer on different distributions over the training samples. However, since most of the boosting algorithms are designed for the classification problem that is very different from the translation problem in natural language processing, several key components have to be redesigned when boosting is adapted to SMT system combination.

3.1 Training

In this work, Minimum Error Rate Training (MERT) proposed by Och (2003) is used to estimate feature weights $\boldsymbol{\lambda}$ over a series of training samples. As in other state-of-the-art SMT systems, BLEU is selected as the accuracy measure to define the error function used in MERT. Since the weights of training samples are not taken into account in BLEU[2], we modify the original definition of BLEU to make it sensitive to the distribution $D_t(i)$ over the training samples. The modified version of BLEU is called *weighted BLEU* (WBLEU) in this paper.

Let $E = e_1 \ldots e_m$ be the translations produced by the system, $R = \boldsymbol{r}_1 \ldots \boldsymbol{r}_m$ be the reference translations where $\boldsymbol{r}_i = \{r_{i1}, \ldots, r_{iN}\}$, and $D_t(i)$ be the weight of the i-th training sample $(f_i, \boldsymbol{r}_i)$. The weighted BLEU metric has the following form:

$$\begin{aligned}&\mathrm{WBLEU}(E,R)\\ &= \exp\left(1-\max\left\{1,\frac{\sum_{i=1}^{m} D_t(i)\min_{1\le j\le N}\{|\,g_1(r_{ij})\,|\}}{\sum_{i=1}^{m} D_t(i)\,|\,g_1(e_i)\,|}\right\}\right)\times\\ &\prod_{n=1}^{4}\left(\frac{\sum_{i=1}^{\mathrm{m}} D_t(i)\left|g_n(e_i)\bigcap\left(\bigcup_{j=1}^{N} g_n(r_{ij})\right)\right|}{\sum_{i=1}^{\mathrm{m}} D_t(i)\left|g_n(e_i)\right|}\right)^{1/4}\end{aligned} \quad (5)$$

where $g_n(s)$ is the multi-set of all n-grams in a string s. In this definition, n-grams in e_i and $\{r_{ij}\}$ are weighted by $D_t(i)$. If the i-th training sample has a larger weight, the corresponding n-grams will have more contributions to the overall score $\mathrm{WBLEU}(E,R)$. As a result, the i-th training sample gains more importance in MERT. Obviously the original BLEU is just a special case of WBLEU when all the training samples are equally weighted.

As the weighted BLEU is used to measure the translation accuracy on the training set, the error rate is defined to be:

$$\varepsilon_t = 1 - \mathrm{WBLEU}(E,R) \quad (6)$$

3.2 Re-weighting

Another key point is the maintaining of the distribution $D_t(i)$ over the training set. Initially all the weights of training samples are set equally. On each round, we increase the weights of the samples that are relatively poorly translated by the current weak system so that the MERT-based trainer can focus on the hard samples in next round. The update rule is given in Equation 4 with two parameters α_t and l_i in it.

α_t can be regarded as a measure of the importance that the t-th weak system gains in boosting. The definition of α_t guarantees that α_t always has a positive value[3]. A main effect of α_t is to scale the weight updating (e.g. a larger α_t means a greater update).

l_i is the loss on the i-th sample. For each i, let $\{e_{i1}, \ldots, e_{in}\}$ be the n-best translation candidates produced by the system. The loss function is defined to be:

$$l_i = \mathrm{BLEU}(e_i^*, \boldsymbol{r}_i) - \frac{1}{k}\sum_{j=1}^{k}\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i) \quad (7)$$

where $\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i)$ is the smoothed sentence-level BLEU score (Liang et al., 2006) of the translation e with respect to the reference translations $\boldsymbol{r}_i$, and e_i^* is the oracle translation which is selected from $\{e_{i1}, \ldots, e_{in}\}$ in terms of $\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i)$. l_i can be viewed as a measure of the average cost that we guess the top-k translation candidates instead of the oracle translation. The value of l_i counts for the magnitude of weight update, that is, a larger l_i means a larger weight update on $D_t(i)$. The definition of the loss function here is similar to the one used in (Chiang et al., 2008) where only the top-1 translation candidate (i.e. $k = 1$) is taken into account.

3.3 System Combination Scheme

In the last step of our method, a strong translation system $v(u(\boldsymbol{\lambda}^*_1), \ldots, u(\boldsymbol{\lambda}^*_T))$ is built from the

[2] In this paper, we use the NIST definition of BLEU where the *effective reference length* is the length of the shortest reference translation.

[3] Note that the definition of α_t here is different from that in the original AdaBoost algorithm (Freund and Schapire, 1997; Schapire, 2001) where α_t is a negative number when $\varepsilon_t > 0.5$.

ensemble of member systems $\{u(\boldsymbol{\lambda}^*_1), ..., u(\boldsymbol{\lambda}^*_T)\}$. In this work, a sentence-level combination method is used to select the best translation from the pool of the *n*-best outputs of all the member systems.

Let $H(u(\boldsymbol{\lambda}^*_t))$ (or H_t for short) be the set of the *n*-best translation candidates produced by the *t*-th member system $u(\boldsymbol{\lambda}^*_t)$, and $H(v)$ be the union set of all H_t (i.e. $H(v) = \bigcup H_t$). The final translation is generated from $H(v)$ based on the following scoring function:

$$e^* = \arg\max_{e \in H(v)} \sum_{t=1}^{T} \beta_t \cdot \phi_t(e) + \psi(e, H(v)) \quad (8)$$

where $\phi_t(e)$ is the log-scaled model score of *e* in the *t*-th member system, and β_t is the corresponding feature weight. It should be noted that $e \in H_i$ may not exist in any $H_{i' \neq i}$. In this case, we can still calculate the model score of *e* in any other member systems, since all the member systems are based on the same model and share the same feature space. $\psi(e, H(v))$ is a consensus-based scoring function which has been successfully adopted in SMT system combination (Duan et al., 2009; Hildebrand and Vogel, 2008; Li et al., 2009). The computation of $\psi(e, H(v))$ is based on a linear combination of a set of *n*-gram consensuses-based features.

$$\psi(e, H(v)) = \sum_n \theta_n^+ \cdot h_n^+(e, H(v)) + \sum_n \theta_n^- \cdot h_n^-(e, H(v)) \quad (9)$$

For each order of *n*-gram, $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$ are defined to measure the *n*-gram agreement and disagreement between *e* and other translation candidates in $H(v)$, respectively. θ_n^+ and θ_n^- are the feature weights corresponding to $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$. As $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$ used in our work are exactly the same as the features used in (Duan et al., 2009) and similar to the features used in (Hildebrand and Vogel, 2008; Li et al., 2009), we do not present the detailed description of them in this paper.

If *p* orders of *n*-gram are used in computing $\psi(e, H(v))$, the total number of features in the system combination will be $T + 2 \times p$ (*T* model-score-based features defined in Equation 8 and $2 \times p$ consensus-based features defined in Equation 9). Since all these features are combined linearly, we use MERT to optimize them for the combination model.

4 Optimization

If implemented naively, the translation speed of the final translation system will be very slow. For a given input sentence, each member system has to encode it individually, and the translation speed is inversely proportional to the number of member systems generated by our method. Fortunately, with the thought of computation, there are a number of optimizations that can make the system much more efficient in practice.

A simple solution is to run member systems in parallel when translating a new sentence. Since all the member systems share the same data resources, such as language model and translation table, we only need to keep one copy of the required resources in memory. The translation speed just depends on the computing power of parallel computation environment, such as the number of CPUs.

Furthermore, we can use joint decoding techniques to save the computation of the equivalent translation hypotheses among member systems. In joint decoding of member systems, the search space is structured as a translation hypergraph where the member systems can share their translation hypotheses. If more than one member systems share the same translation hypothesis, we just need to compute the corresponding feature values only once, instead of repeating the computation in individual decoders. In our experiments, we find that over 60% translation hypotheses can be shared among member systems when the number of member systems is over 4. This result indicates that promising speed improvement can be achieved by using the joint decoding and hypothesis sharing techniques.

Another method to speed up the system is to accelerate *n*-gram language model with *n*-gram caching techniques. In this method, a *n*-gram cache is used to store the most frequently and recently accessed *n*-grams. When a new *n*-gram is accessed during decoding, the cache is checked first. If the required *n*-gram hits the cache, the corresponding *n*-gram probability is returned by the cached copy rather than refetching the original data in language model. As the translation speed of SMT system depends heavily on the computation of *n*-gram language model, the acceleration of *n*-gram language model generally leads to substantial speed-up of SMT system. In our implementation, the *n*-gram caching in general brings us over 30% speed improvement of the system.

5 Experiments

Our experiments are conducted on Chinese-to-English translation in three SMT systems.

5.1 Baseline Systems

The first SMT system is a phrase-based system with two reordering models including the maximum entropy-based lexicalized reordering model proposed by Xiong et al. (2006) and the hierarchical phrase reordering model proposed by Galley and Manning (2008). In this system all phrase pairs are limited to have source length of at most 3, and the reordering limit is set to 8 by default[4].

The second SMT system is an in-house reimplementation of the Hiero system which is based on the hierarchical phrase-based model proposed by Chiang (2005).

The third SMT system is a syntax-based system based on the string-to-tree model (Galley et al., 2006; Marcu et al., 2006), where both the minimal GHKM and SPMT rules are extracted from the bilingual text, and the composed rules are generated by combining two or three minimal GHKM and SPMT rules. Synchronous binarization (Zhang et al., 2006; Xiao et al., 2009) is performed on each translation rule for the CKY-style decoding.

In this work, baseline system refers to the system produced by the boosting-based system combination when the number of iterations (i.e. T) is set to 1. To obtain satisfactory baseline performance, we train each SMT system for 5 times using MERT with different initial values of feature weights to generate a group of *baseline candidates*, and then select the best-performing one from this group as the final baseline system (i.e. the starting point in the boosting process) for the following experiments.

5.2 Experimental Setup

Our bilingual data consists of 140K sentence pairs in the FBIS data set[5]. GIZA++ is employed to perform the bi-directional word alignment between the source and target sentences, and the final word alignment is generated using the intersect-diag-grow method. All the word-aligned bilingual sentence pairs are used to extract phrases and rules for the baseline systems. A 5-gram language model is trained on the target-side of the bilingual data and the Xinhua portion of English Gigaword corpus. Berkeley Parser is used to generate the English parse trees for the rule extraction of the syntax-based system. The data set used for weight training in boosting-based system combination comes from NIST MT03 evaluation set. To speed up MERT, all the sentences with more than 20 Chinese words are removed. The test sets are the NIST evaluation sets of MT04, MT05 and MT06. The translation quality is evaluated in terms of case-insensitive NIST version BLEU metric. Statistical significant test is conducted using the bootstrap re-sampling method proposed by Koehn (2004).

Beam search and cube pruning (Huang and Chiang, 2007) are used to prune the search space in all the three baseline systems. By default, both of the beam size and the size of n-best list are set to 20.

In the settings of boosting-based system combination, the maximum number of iterations is set to 30, and k (in Equation 7) is set to 5. The n-gram consensuses-based features (in Equation 9) used in system combination ranges from unigram to 4-gram.

5.3 Evaluation of Translations

First we investigate the effectiveness of the boosting-based system combination on the three systems.

Figures 2-5 show the BLEU curves on the development and test sets, where the X-axis is the iteration number, and the Y-axis is the BLEU score of the system generated by the boosting-based system combination. The points at iteration 1 stand for the performance of the baseline systems. We see, first of all, that all the three systems are improved during iterations on the development set. This trend also holds on the test sets. After 5, 7 and 8 iterations, relatively stable improvements are achieved by the phrase-based system, the Hiero system and the syntax-based system, respectively. The BLEU scores tend to converge to the stable values after 20 iterations for all the systems. Figures 2-5 also show that the boosting-based system combination seems to be more helpful to the phrase-based system than to the Hiero system and the syntax-based system. For the phrase-based system, it yields over 0.6 BLEU point gains just after the 3rd iteration on all the data sets.

Table 1 summarizes the evaluation results, where the BLEU scores at iteration 5, 10, 15, 20 and 30 are reported for the comparison. We see that the boosting-based system method stably ac-

[4] Our in-house experimental results show that this system performs slightly better than Moses on Chinese-to-English translation tasks.

[5] LDC catalog number: LDC2003E14

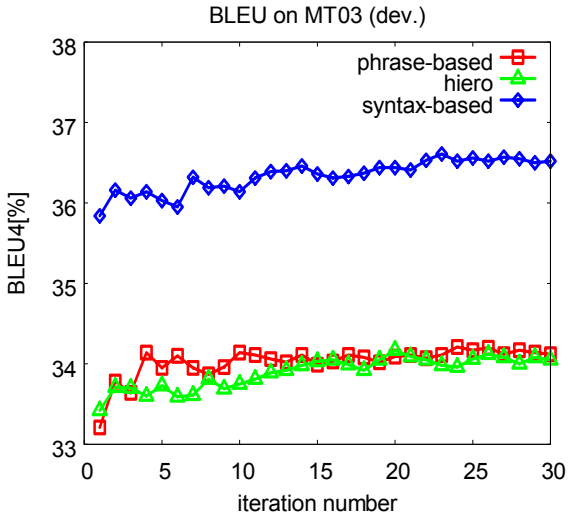


Figure 2: BLEU scores on the development set

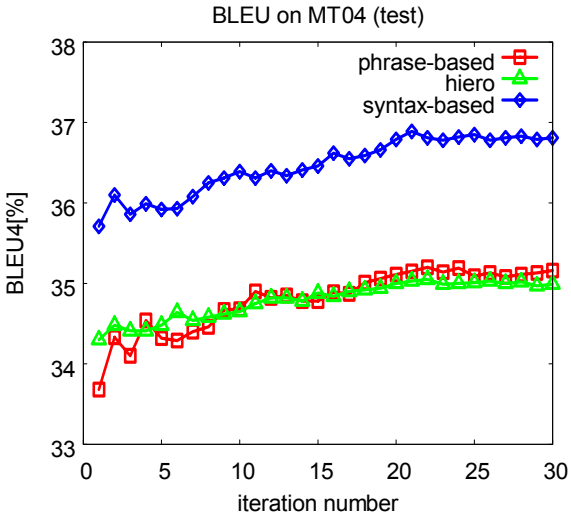


Figure 3: BLEU scores on the test set of MT04

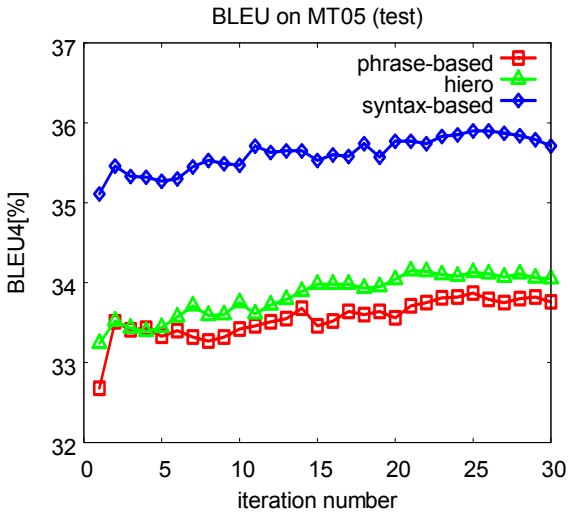


Figure 4: BLEU scores on the test set of MT05

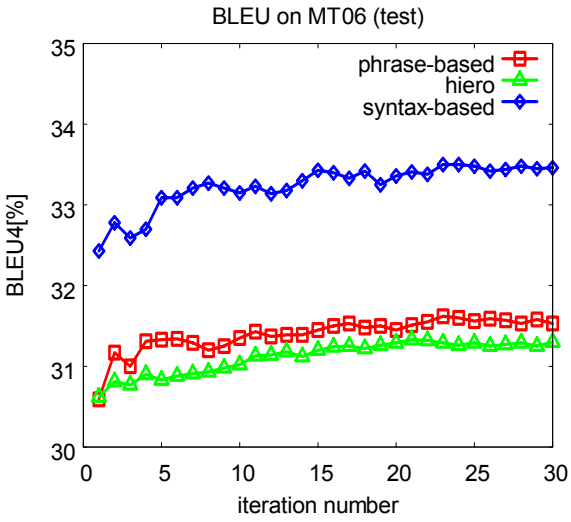


Figure 5: BLEU scores on the test set of MT06

	Phrase-based				Hiero				Syntax-based			
	Dev.	MT04	MT05	MT06	Dev.	MT04	MT05	MT06	Dev.	MT04	MT05	MT06
Baseline	33.21	33.68	32.68	30.59	33.42	34.30	33.24	30.62	35.84	35.71	35.11	32.43
Baseline+600best	33.32	33.93	32.84	30.76	33.48	34.46	33.39	30.75	35.95	35.88	35.23	32.58
Boosting-5Iterations	33.95*	34.32*	33.33*	31.33*	33.73	34.48	33.44	30.83	36.03	35.92	35.27	33.09
Boosting-10Iterations	34.14*	34.68*	33.42*	31.35*	33.75	34.65	33.75*	31.02	36.14	36.39*	35.47	33.15*
Boosting-15Iterations	33.99*	34.78*	33.46*	31.45*	34.03*	34.88*	33.98*	31.20*	36.36*	36.46*	35.53*	33.43*
Boosting-20Iterations	34.09*	35.11*	33.56*	31.45*	34.17*	35.00*	34.04*	31.29*	36.44*	36.79*	35.77*	33.36*
Boosting-30Iterations	34.12*	35.16*	33.76*	31.59*	34.05*	34.99*	34.05*	31.30*	36.52*	36.81*	35.71*	33.46*

Table 1: Summary of the results (BLEU4[%]) on the development and test sets. * = significantly better than baseline ($p < 0.05$).

hieves significant BLEU improvements after 15 iterations, and the highest BLEU scores are generally yielded after 20 iterations.

Also as shown in Table 1, over 0.7 BLEU point gains are obtained on the phrase-based system after 10 iterations. The largest BLEU improvement on the phrase-based system is over 1 BLEU point in most cases. These results reflect that our method is relatively more effective for the phrase-based system than for the other two systems, and thus confirms the fact we observed in Figures 2-5.

We also investigate the impact of *n*-best list size on the performance of baseline systems. For the comparison, we show the performance of the baseline systems with the *n*-best list size of 600 (*Baseline+600best* in Table 1) which equals to the maximum number of translation candidates accessed in the final combination system (combine 30 member systems, i.e. *Boosing-30Iterations*).

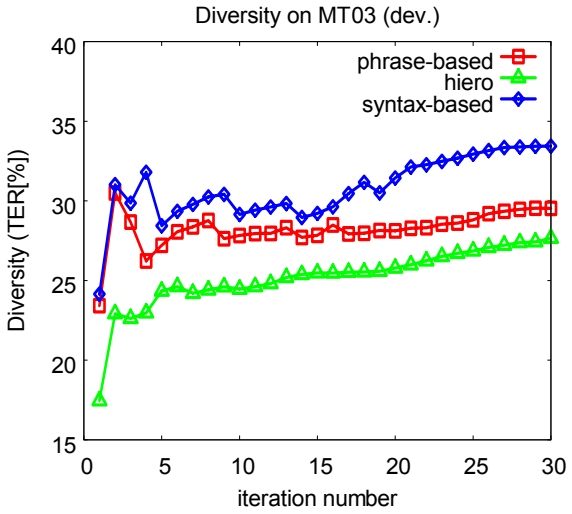


Figure 6: Diversity on the development set

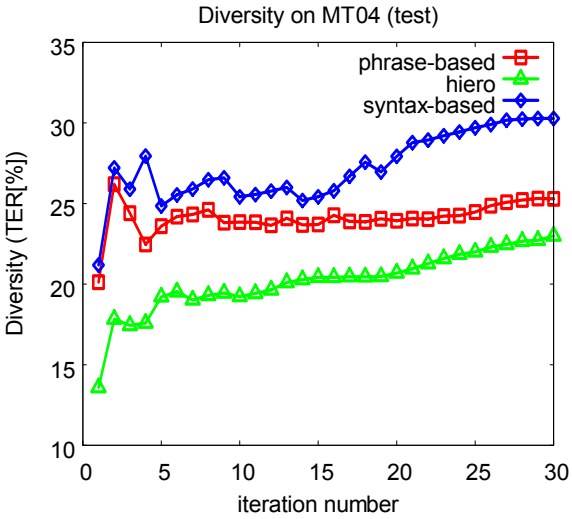


Figure 7: Diversity on the test set of MT04

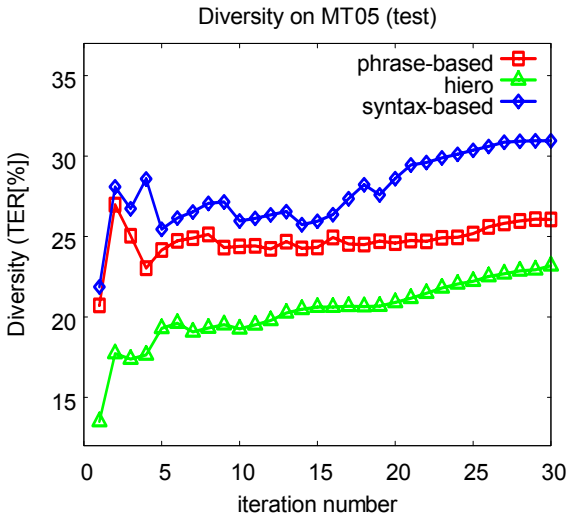


Figure 8: Diversity on the test set of MT05

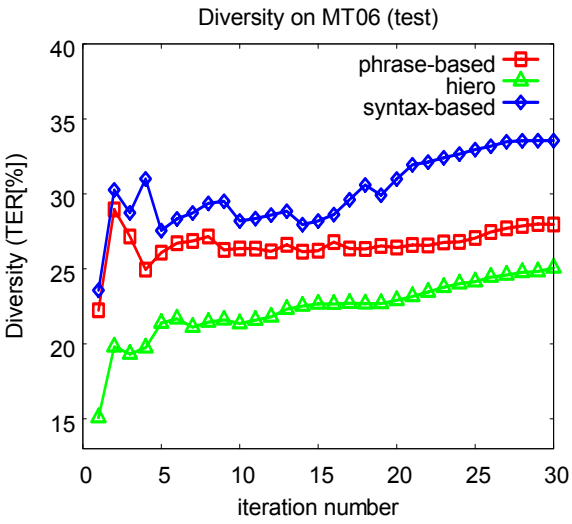


Figure 9: Diversity on the test set of MT06

As shown in Table 1, *Baseline+600best* obtains stable improvements over *Baseline*. It indicates that the access to larger *n*-best lists is helpful to improve the performance of baseline systems. However, the improvements achieved by *Baseline+600best* are modest compared to the improvements achieved by *Boosting-30Iterations*. These results indicate that the SMT systems can benefit more from the diversified outputs of member systems rather than from larger *n*-best lists produced by a single system.

5.4 Diversity among Member Systems

We also study the change of diversity among the outputs of member systems during iterations. The diversity is measured in terms of the Translation Error Rate (TER) metric proposed in (Snover et al., 2006). A higher TER score means that more edit operations are performed if we transform one translation output into another translation output, and thus reflects a larger diversity between the two outputs. In this work, the TER score for a given group of member systems is calculated by averaging the TER scores between the outputs of each pair of member systems in this group.

Figures 6-9 show the curves of diversity on the development and test sets, where the X-axis is the iteration number, and the Y-axis is the diversity. The points at iteration 1 stand for the diversities of baseline systems. In this work, the baseline's diversity is the TER score of the group of baseline candidates that are generated in advance (Section 5.1).

We see that the diversities of all the systems increase during iterations in most cases, though a few drops occur at a few points. It indicates that our method is very effective to generate diversified member systems. In addition, the diversities of baseline systems (iteration 1) are much lower

than those of the systems generated by boosting (iterations 2-30). Together with the results shown in Figures 2-5, it confirms our motivation that the diversified translation outputs can lead to performance improvements over the baseline systems.

Also as shown in Figures 6-9, the diversity of the Hiero system is much lower than that of the phrase-based and syntax-based systems at each individual setting of iteration number. This interesting finding supports the observation that the performance of the Hiero system is relatively more stable than the other two systems as shown in Figures 2-5. The relative lack of diversity in the Hiero system might be due to the *spurious ambiguity* in Hiero derivations which generally results in very few different translations in translation outputs (Chiang, 2007).

5.5 Evaluation of Oracle Translations

In this set of experiments, we evaluate the oracle performance on the *n*-best lists of the baseline systems and the combined systems generated by boosting-based system combination. Our primary goal here is to study the impact of our method on the upper-bound performance.

Table 2 shows the results, where *Baseline+600best* stands for the top-600 translation candidates generated by the baseline systems, and *Boosting-30iterations* stands for the ensemble of 30 member systems' top-20 translation candidates. As expected, the oracle performance of *Boosting-30Iterations* is significantly higher than that of *Baseline+600best*. This result indicates that our method can provide much "better" translation candidates for system combination than enlarging the size of *n*-best list naively. It also gives us a rational explanation for the significant improvements achieved by our method as shown in Section 5.3.

Data Set	Method	Phrase-based	Hiero	Syntax-based
Dev.	Baseline+600best	46.36	46.51	46.92
	Boosting-30Iterations	47.78*	47.44*	48.70*
MT04	Baseline+600best	43.94	44.52	46.88
	Boosting-30Iterations	45.97*	45.47*	49.40*
MT05	Baseline+600best	42.32	42.47	45.21
	Boosting-30Iterations	44.82*	43.44*	47.02*
MT06	Baseline+600best	39.47	39.39	40.52
	Boosting-30Iterations	41.51*	40.10*	41.88*

Table 2: Oracle performance of various systems. * = significantly better than baseline ($p < 0.05$).

6 Related Work

Boosting is a machine learning (ML) method that has been well studied in the ML community (Freund, 1995; Freund and Schapire, 1997; Collins et al., 2002; Rudin et al., 2007), and has been successfully adopted in natural language processing (NLP) applications, such as document classification (Schapire and Singer, 2000) and named entity classification (Collins and Singer, 1999). However, most of the previous work did not study the issue of how to improve a single SMT engine using boosting algorithms. To our knowledge, the only work addressing this issue is (Lagarda and Casacuberta, 2008) in which the boosting algorithm was adopted in phrase-based SMT. However, Lagarda and Casacuberta (2008)'s method calculated errors over the phrases that were chosen by phrase-based systems, and could not be applied to many other SMT systems, such as hierarchical phrase-based systems and syntax-based systems. Differing from Lagarda and Casacuberta's work, we are concerned more with proposing a general framework which can work with most of the current SMT models and empirically demonstrating its effectiveness on various SMT systems.

There are also some other studies on building diverse translation systems from a single translation engine for system combination. The first attempt is (Macherey and Och, 2007). They empirically showed that diverse translation systems could be generated by changing parameters at early-stages of the training procedure. Following Macherey and Och (2007)'s work, Duan et al. (2009) proposed a feature subspace method to build a group of translation systems from various different sub-models of an existing SMT system. However, Duan et al. (2009)'s method relied on the heuristics used in feature sub-space selection. For example, they used the remove-one-feature strategy and varied the order of *n*-gram language model to obtain a satisfactory group of diverse systems. Compared to Duan et al. (2009)'s method, a main advantage of our method is that it can be applied to most of the SMT systems without designing any heuristics to adapt it to the specified systems.

7 Discussion and Future Work

Actually the method presented in this paper is doing something rather similar to Minimum Bayes Risk (MBR) methods. A main difference lies in that the consensus-based combination method here does not model the posterior probability of each hypothesis (i.e. all the hypotheses are assigned an equal posterior probability when we calculate the consensus-based features).

Greater improvements are expected if MBR methods are used and consensus-based combination techniques smooth over noise in the MERT pipeline.

In this work, we use a sentence-level system combination method to generate final translations. It is worth studying other more sophisticated alternatives, such as word-level and phrase-level system combination, to further improve the system performance.

Another issue is how to determine an appropriate number of iterations for boosting-based system combination. It is especially important when our method is applied in the real-world applications. Our empirical study shows that the stable and satisfactory improvements can be achieved after 6-8 iterations, while the largest improvements can be achieved after 20 iterations. In our future work, we will study in-depth principled ways to determine the appropriate number of iterations for boosting-based system combination.

8 Conclusions

We have proposed a boosting-based system combination method to address the issue of building a strong translation system from a group of weak translation systems generated from a single SMT engine. We apply our method to three state-of-the-art SMT systems, and conduct experiments on three NIST Chinese-to-English MT evaluations test sets. The experimental results show that our method is very effective to improve the translation accuracy of the SMT systems.

Acknowledgements

This work was supported in part by the National Science Foundation of China (60873091) and the Fundamental Research Funds for the Central Universities (N090604008). The authors would like to thank the anonymous reviewers for their pertinent comments, Tongran Liu, Chunliang Zhang and Shujie Yao for their valuable suggestions for improving this paper, and Tianning Li and Rushan Chen for developing parts of the baseline systems.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of* ACL 2005, Ann Arbor, Michigan, pages 263-270.

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

David Chiang, Yuval Marton and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proc. of* EMNLP 2008, Honolulu, pages 224-233.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proc. of* EMNLP/VLC 1999, pages 100-110.

Michael Collins, Robert Schapire and Yoram Singer. 2002. Logistic Regression, AdaBoost and Bregman Distances. *Machine Learning*, 48(3): 253-285.

Brooke Cowan, Ivona Kučerová and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. of* EMNLP 2006, pages 232-241.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of* ACL 2005, Ann Arbor, Michigan, pages 541-548.

Nan Duan, Mu Li, Tong Xiao and Ming Zhou. 2009. The Feature Subspace Method for SMT System Combination. In *Proc. of* EMNLP 2009, pages 1096-1104.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of* ACL 2003, pages 205-208.

Yoav Freund. 1995. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2): 256-285.

Yoav Freund and Robert Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. of* ACL 2006, Sydney, Australia, pages 961-968.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of* EMNLP 2008, Hawaii, pages 848-856.

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proc. of* the 8th AMTA conference, pages 254-261.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of* ACL 2007, Prague, Czech Republic, pages 144-151.

Philipp Koehn, Franz Och and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of* HLT-NAACL 2003, Edmonton, USA, pages 48-54.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of* EMNLP 2004, Barcelona, Spain, pages 388-395.

Antonio Lagarda and Francisco Casacuberta. 2008. Applying Boosting to Statistical Machine Translation. In *Proc. of* the 12th EAMT conference, pages 88-96.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li and Ming Zhou. 2009. Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders. In *Proc. of* ACL-IJCNLP 2009, Singapore, pages 585-592.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of* COLING/ACL 2006, pages 104-111.

Yang Liu, Qun Liu and Shouxun Lin. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proc. of* ACL 2006, pages 609-616.

Wolfgang Macherey and Franz Och. 2007. An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In *Proc. of* EMNLP 2007, pages 986-995.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of* EMNLP 2006, Sydney, Australia, pages 44-52.

Evgeny Matusov, Nicola Ueffing and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of* EACL 2006, pages 33-40.

Franz Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. of* ACL 2002, Philadelphia, pages 295-302.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of* ACL 2003, Japan, pages 160-167.

Antti-Veikko Rosti, Spyros Matsoukas and Richard Schwartz. 2007. Improved Word-Level System Combination for Machine Translation. In *Proc. of* ACL 2007, pages 312-319.

Cynthia Rudin, Robert Schapire and Ingrid Daubechies. 2007. Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 35(6): 2723-2768.

Robert Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135-168.

Robert Schapire. The boosting approach to machine learning: an overview. 2001. In *Proc. of* MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, USA, pages 1-23.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proc. of* the 7th AMTA conference, pages 223-231.

Tong Xiao, Mu Li, Dongdong Zhang, Jingbo Zhu and Ming Zhou. 2009. Better Synchronous Binarization for Machine Translation. In *Proc. of* EMNLP 2009, Singapore, pages 362-370.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proc. of* ACL 2006, Sydney, pages 521-528.

Hao Zhang, Liang Huang, Daniel Gildea and Kevin Knight. 2006. Synchronous Binarization for Machine Translation. In *Proc. of* HLT-NAACL 2006, New York, USA, pages 256- 263.

Fine-grained Genre Classification using Structural Learning Algorithms

Zhili Wu
Centre for Translation Studies
University of Leeds, UK
z.wu@leeds.ac.uk

Katja Markert
School of Computing
University of Leeds, UK
scskm@leeds.ac.uk

Serge Sharoff
Centre for Translation Studies
University of Leeds, UK
s.sharoff@leeds.ac.uk

Abstract

Prior use of machine learning in genre classification used a list of labels as classification categories. However, genre classes are often organised into hierarchies, e.g., covering the subgenres of fiction. In this paper we present a method of using the hierarchy of labels to improve the classification accuracy. As a testbed for this approach we use the Brown Corpus as well as a range of other corpora, including the BNC, HGC and Syracuse. The results are not encouraging: apart from the Brown corpus, the improvements of our structural classifier over the flat one are not statistically significant. We discuss the relation between structural learning performance and the visual and distributional balance of the label hierarchy, suggesting that only balanced hierarchies might profit from structural learning.

1 Introduction

Automatic genre identification (AGI) can be traced to the mid-1990s (Karlgren and Cutting, 1994; Kessler et al., 1997), but this research became much more active in recent years, partly because of the explosive growth of the Web, and partly because of the importance of making genre distinctions in NLP applications. In Information Retrieval, given the large number of web pages on any given topic, it is often difficult for the users to find relevant pages that are in the right genre (Vidulin et al., 2007). As for other applications, the accuracy of many tasks, such as machine translation, POS tagging (Giesbrecht and Evert, 2009) or identification of discourse relations (Webber, 2009) relies of defining the language model suitable for the genre of a given text. For example, the accuracy of POS tagging reaching 96.9% on newspaper texts drops down to 85.7% on forums (Giesbrecht and Evert, 2009), i.e., every seventh word in forums is tagged incorrectly.

This interest in genres resulted in a proliferation of studies on corpus development of web genres and comparison of methods for AGI. The two corpora commonly used for this task are KI-04 (Meyer zu Eissen and Stein, 2004) and Santinis (Santini, 2007). The best results reported for these corpora (with 10-fold cross-validation) reach 84.1% on KI-04 and 96.5% accuracy on Santinis (Kanaris and Stamatatos, 2009). In our research (Sharoff et al., 2010) we produced even better results on these two benchmarks (85.8% and 97.1%, respectively). However, this impressive accuracy is not realistic *in vivo*, i.e., in classifying web pages retrieved as a result of actual queries. One reason comes from the limited number of genres present in these two collections (eight genres in KI-04 and seven in Santinis). As an example, only front pages of online newspapers are listed in Santinis, but not actual newspaper articles, so once an article is retrieved, it cannot be assigned to any class at all. Another reason why the high accuracy is not useful concerns the limited number of sources in each collection, e.g., all FAQs in Santinis come from either a website with FAQs on hurricanes or another one with tax advice. In the end, a classifier built for FAQs on this training data relies on a high topic-genre correlation in this particular collection and fails to spot any other FAQs.

There are other corpora, which are more diverse in the range of their genres, such as the fifteen genres of the Brown Corpus (Kučera and Francis, 1967) or the seventy genres of the BNC (Lee, 2001), but because of the number of genres in them and the diversity of documents within each genre, the accuracy of prior work on these collections is much less impressive. For example, Karlgren and Cutting (1994) using linear discriminant analysis achieve an accuracy of 52% without us-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 749–759,
Uppsala, Sweden, 11-16 July 2010.

ing cross-validation (the entire Brown Corpus was used as both the test set and training set), with the accuracy improving to 65% when the 15 genres are collapsed into 10, and to 73% with only 4 genres (Figure 1). This result suggests the importance of the hierarchy of genres. Firstly, making a decision on higher levels might be easier than on lower levels (fiction or non-fiction rather than science fiction or mystery). Secondly, we might be able to improve the accuracy on lower levels, by taking into account the relevant position of each node in the hierarchy (distinguishing between `reportage` or `editorial` becomes easier when we know they are safely under the category of `press`).

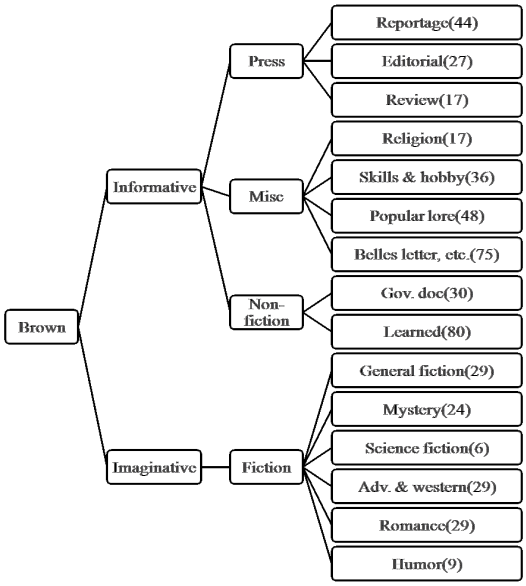


Figure 1: Hierarchy of Brown corpus.

This paper explores a way of using information on the hierarchy of labels for improving fine-grained genre classification. To the best of our knowledge, this is the first work presenting structural genre classification and distance measures for genres. In Section 2 we present a structural reformulation of Support Vector Machines (SVMs) that can take similarities between different genres into account. This formulation necessitates the development of distance measures between different genres in a hierarchy, of which we present three different types in Section 3, along with possible estimation procedures for these distances. We present experiments with these novel structural SVMs and distance measures on three different corpora in Section 4. Our experiments show that structural SVMs can outperform the non-structural standard. However, the improvement is only statistically significant on the Brown corpus. In Section 5 we investigate potential reasons for this, including the (im)balance of different genre hierarchies and problems with our distance measures.

2 Structural SVMs

Discriminative methods are often used for classification, with SVMs being a well-performing method in many tasks (Boser et al., 1992; Joachims, 1999). Linear SVMs on a flat list of labels achieve high efficiency and accuracy in text classification when compared to nonlinear SVMs or other state-of-the-art methods. As for structural output learning, a few SVM-based objective functions have been proposed, including margin formulation for hierarchical learning (Dekel et al., 2004) or general structural learning (Joachims et al., 2009; Tsochantaridis et al., 2005). But many implementations are not publicly available, and their scalability to real-life text classification tasks is unknown. Also they have not been applied to genre classification.

Our formulation can be taken as a special instance of the structural learning framework in (Tsochantaridis et al., 2005). However, they concentrate on more complicated label structures as for sequence alignment or parsing. They proposed two formulations, slack-rescaling and margin-rescaling, claiming that margin-rescaling has two disadvantages. First, it potentially gives significant weight to output values that might not be easily confused with the target values, because every increase in the loss increases the required margin. However, they did not provide empirical evidence for this claim. Second, margin rescaling is not necessarily invariant to the scaling of the distance matrix. We still used margin-rescaling because it allows us to use the sequential dual method for large-scale implementation (Keerthi et al., 2008), which is not applicable to the slack-rescaling formulation. For web page classification we will need fast processing. In addition, we performed model calibration to address the second disadvantage (distance matrix invariance).

Let $\mathbf{x}$ be a document and $\mathbf{w}_m$ a weight vector associated with the genre class m in a corpus with k genres at the most fine-grained level. The predicted class is the class achieving the maximum inner product between $\mathbf{x}$ and the weight vector for the class, denoted as,

$$\arg\max_m \ \mathbf{w}_m^T \mathbf{x}, \forall m. \qquad (1)$$

Accurate prediction requires that when a document vector is multiplied with the weight vector associated with its own class, the resulting inner product should be larger than its inner products with a weight vector for any other genre class m. This helps us to define criteria for weight vectors. Let $\mathbf{x}_i$ be the $i-$th training document, and y_i its genre label. For its weight vector $\mathbf{w}_{y_i}$, the inner product $\mathbf{w}_{y_i}^T \mathbf{x}_i$ should be larger than all other products $\mathbf{w}_m^T \mathbf{x}_i$, that is,

$$\mathbf{w}_{y_i}^T \mathbf{x}_i - \mathbf{w}_m^T \mathbf{x}_i \geq 0, \forall m. \quad (2)$$

To strengthen the constraints, the zero value on the right hand side of the inequality for the flat SVM can be replaced by a positive value, corresponding to a distance measure $h(y_i, m)$ between two genre classes, leading to the following constraint:

$$\mathbf{w}_{y_i}^T \mathbf{x}_i - \mathbf{w}_m^T \mathbf{x}_i \geq h(y_i, m), \forall m. \quad (3)$$

To allow feasible models, in real scenarios such constraints can be violated, but the degree of violation is expected to be small. For each document, the maximum violation in the k constraints is of interest, as given by the following loss term:

$$Loss_i = \max_m \{h(y_i, m) - \mathbf{w}_{y_i}^T \mathbf{x}_i + \mathbf{w}_m^T \mathbf{x}_i\}. \quad (4)$$

Adding up all loss terms over all training documents, and further introducing a term to penalize large values in the weight vectors, we have the following objective function (C is a user-specified nonnegative parameter).

$$\min_{m,i} : \frac{1}{2} \sum_{m=1}^{k} \mathbf{w}_m^T \mathbf{w}_m + C \sum_{i=1}^{p} Loss_i. \quad (5)$$

Efficient methods can be derived by borrowing the sequential dual methods in (Keerthi et al., 2008) or other optimization techniques (Crammer and Singer, 2002).

3 Genre Distance Measures

The structural SVM (Section 2) requires a distance measure h between two genres. We can derive such distance measures from the genre hierarchy in a way similar to word similarity measures that were invented for lexical hierarchies such as WordNet (see (Pedersen et al., 2007) for an overview). In the following, we will first shortly summarise path-based and information-based measures for similarity. However, information-based measures are based on the information content of a node in a hierarchy. Whereas the information content of a word or concept in a lexical hierarchy has been well-defined (Resnik, 1995), it is less clear how to estimate the information content of a genre label. We will therefore discuss several different ways of estimating information content of nodes in a genre hierarchy.

3.1 Distance Measures based on Path Length

If genre labels are organised into a tree (Figure 1), one of the simplest ways to measure distance between two genre labels (= tree nodes) is ***path length*** ($h(a,b)_{plen}$):

$$f(a, LCS(a,b)) + f(b, LCS(a,b)), \quad (6)$$

where a and b are two nodes in the tree, $LCS(a,b)$ is their Least Common Subsumer, and $f(a, LCS(a,b))$ is the number of levels passed through when traversing from a to the ancestral node $LCS(a,b)$. In other words, the distance counts the number of edges traversed from nodes a to b in the tree. For example, the distance between `Learned` and `Misc` in Figure 1 would be 3.

As an alternative, the ***maximum path length*** $h(a,b)_{pmax}$ to their least common subsumer can be used to reduce the range of possible values:

$$\max\{f(a, LCS(a,b)), f(b, LCS(a,b))\}. \quad (7)$$

The ***Leacock & Chodorow*** similarity measure (Leacock and Chodorow, 1998) normalizes the path length measure (6) by the maximum number of nodes D when traversing down from the root.

$$s(a,b)_{plsk} = -log((h(a,b)_{plen} + 1)/2D). \quad (8)$$

To convert it into a distance measure, we can invert it $h(a,b)_{plsk} = 1/s(a,b)_{plsk}$.

Other path-length based measures include the ***Wu & Palmer*** Similarity (Wu and Palmer, 1994).

$$s(a,b)_{pwupal} = \frac{2f(R, LCS(a,b))}{(f(R,a) + f(R,b))}, \quad (9)$$

where R describes the hierarchy's root node. Here similarity is proportional to the shared path from the root to the least common subsumer of two nodes. Since the Wu & Palmer similarity is always between [0 1), we can convert it into a distance measure by $h(a,b)_{pwupal} = 1 - s(a,b)_{pwupal}$.

3.2 Distance Measures based on Information Content

Path-based distance measures work relatively well on balanced hierarchies such as the one in Figure 1 but fail to treat hierarchies with different levels of granularity well. For lexical hierarchies, as a result, several distance measures based on *information content* have been suggested where the information content of a concept c in a hierarchy is measured by (Resnik, 1995)

$$IC(c) = -log(\frac{freq(c)}{freq(root)}). \quad (10)$$

The frequency $freq$ of a concept c is the sum of the frequency of the node c itself and the frequencies of all its subnodes. Since the root may be a dummy concept, its frequency is simply the sum of the frequencies of all its subnodes. The similarity between two nodes can then be defined as the information content of their least common subsumer:

$$s(a,b)_{resk} = IC(LCS(a,b)). \quad (11)$$

If two nodes just share the root as their subsumer, their similarity will be zero. To convert 11 into a distance measure, it is possible to add a constant 1 to it before inverting it, as given by

$$h(a,b)_{resk} = 1/(s(a,b)_{resk} + 1). \quad (12)$$

Several other similarity measures have been proposed based on the Resnik similarity such as the one by (Lin, 1998):

$$s(a,b)_{lin} = \frac{2IC(LCS(a,b))}{IC(a) + IC(b)}. \quad (13)$$

Again to avoid the effect of zero similarity when defining the ***Lin's distance*** we use:

$$h(a,b)_{lin} = 1/(s(a,b)_{lin} + 1). \quad (14)$$

(Jiang and Conrath, 1997) directly define ***Jiang's distance*** ($h(a,b)_{jng}$):

$$IC(a) + IC(b) - 2IC(LCS(a,b)). \quad (15)$$

3.2.1 Information Content of Genre Labels

The notion of information content of a genre is not straightforward. We use two ways of measuring the frequency $freq$ of a genre, depending on its interpretation.

Genre Frequency based on Document Occurrence. We can interpret the "frequency" of a genre node simply as the number of all documents belonging to that genre (including any of its subgenres). Unfortunately, there are no estimates for genre frequencies on, for example, a representative sample of web documents. Therefore, we approximate genre frequencies from the document frequencies (*dfs*) in the training sets used in classification. Note that (i) for balanced class distributions this information will not be helpful and (ii) that this is a relatively poor substitute for an estimation on an independent, representative corpus.

Genre Frequency based on Genre Labels. We can also use the labels/names of the genre nodes as the unit of frequency estimation. Then, the frequency of a genre node is the occurrence frequency of its label in a corpus plus the occurrence frequencies of the labels of all its subnodes. Note that there is no direct correspondence between this measure and the document frequency of a genre: measuring the number of times the potential genre label *poem* occurs in a corpus is not in any way equivalent to the number of poems in that corpus. However, the measure is still structurally aware as frequencies of labels of subnodes are included, i.e. a higher level genre label will have higher frequency (and lower information content) than a lower level genre label.[1]

For label frequency estimation, we manually expand any label abbreviations (such as "newsp" for BNC genre labels), delete stop words and function words and then use two search methods. For the search method $word$ we simply search the frequency of the genre label in a corpus, using three different corpora (the BNC, Brown and Google web search). As for the BNC and Brown corpus some labels are very rarely mentioned, we for these two corpora use also a search method $gram$ where all character 5-grams within the genre label are searched for and their frequencies aggregated.

3.3 Terminology

Algorithms are prefixed by the kind of distance measure they employ — IC for Information content and p for path-based). If the measure is infor-

[1] Obviously when using this measure we rely on genre labels which are meaningful in the sense that lower level labels were chosen to be more specific and therefore probably rarer terms in a corpus. The measure could not possibly be useful on a genre hierarchy that would give random names to its genres such as *genre 1*.

mation content based the specific measure is mentioned next, such as lin. The way for measuring genre frequency is indicated last with df for measuring via document frequency and $word/gram$ when measured via frequency of genre labels. If frequencies of genre labels are used, the corpus for counting the occurrence of genre labels is also indicated via *brown*, *bnc* or the Web as estimated by Google hit counts gg. Standard non-structural SVMs are indicated by $flat$.

4 Experiments

4.1 Datasets

We use four genre-annotated corpora for genre classification: the Brown Corpus (Kučera and Francis, 1967), BNC (Lee, 2001), HGC (Stubbe and Ringlstetter, 2007) and Syracuse (Crowston et al., 2009). They have a wide variety of genre labels (from 15 in the Brown corpus to 32 genres in HGC to 70 in the BNC to 292 in Syracuse), and different types of hierarchies.

4.2 Evaluation Measures

We use standard classification accuracy (Acc) on the most fine-grained level of target categories in the genre hierarchy.

In addition, given a structural distance H, misclassifications can be weighted based on the distance measure. This allows us to penalize incorrect predictions which are further away in the hierarchy (such as between government documents and westerns) more than "close" mismatches (such as between science fiction and westerns). Formally, given the classification confusion matrix M then each M_{ab} for $a \neq b$ contains the number of class a documents that are misclassified into class b. To achieve proper normalization in giving weights to misclassified entries, we can redistribute a total weight $k-1$ to each row of H proportionally to its values, where k is the number of genres. That is, given g the row summation of H, we define a weight matrix Q by normalizing the rows of H in a way given by $Q_{ab} = (k-1)h_{ab}/g_a,\ a \neq b$. We further assign a unit value to the diagonal of Q. Then it is possible to construct a structurally-aware measure (S-Acc):

$$\text{S-Acc} = \sum_a M_{aa} / \sum_{a,b} M_{ab} Q_{ab}. \qquad (16)$$

4.3 Experimental Setup

We compare structural SVMs using all path-based and information-content based measures (see also Section 3.3). As a baseline we use the accuracy achieved by a standard "flat" SVM.

We use 10-fold (randomised) cross validation throughout. In each fold, for each genre class 10% of documents are used for testing. For the remaining 90%, a portion of 10% are sampled for parameter tuning, leaving 80% for training. In each round the validation set is used to help determine the best C associated with Equation (5) based on the validation accuracy from the candidate list 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1. Note via this experiment setup, all methods are tuned to their best performance.

For any algorithm comparison, we use a McNemar test with the significance level of 5% as recommended by (Dietterich, 1998).

4.4 Features

The features used for genre classification are character 4-grams for all algorithms, i.e. each document is represented by a binary vector indicating the existence of each character 4-gram. We used character n-grams because they are very easy to extract, language-independent (no need to rely on parsing or even stemming), and they are known to have the best performance in genre classification tasks (Kanaris and Stamatatos, 2009; Sharoff et al., 2010).

4.5 Brown Corpus Results

The Brown Corpus has 500 documents and is organized in a hierarchy with a depth of 3. It contains 15 end-level genres. In one experiment in (Karlgren and Cutting, 1994) the subgenres under *fiction* are grouped together, leading to 10 genres to classify.

Results on 10-genre Brown Corpus. A standard flat SVM achieves an accuracy of 64.4% whereas the best structural SVM based on Lin's information content distance measure (IC-lin-word-bnc) achieves 68.8% accuracy, significantly better at the 1% level. The result is also significantly better than prior work on the Brown corpus in (Karlgren and Cutting, 1994) (who use the whole corpus as test as well as training data). Table 1 summarizes the best performing measures that all outperform the flat SVM at the 1% level.

Table 1: Brown 10-genre Classification Results.

Method	Accuracy
Karlgren and Cutting, 1994	65 (Training)
Flat SVM	64.40
SSVM(IC-lin-word-bnc)	68.80
SSVM(IC-lin-word-br)	68.60
SSVM(IC-lin-gram-br)	67.80

Figure 2 provides the box plots of accuracy scores. The dashed boxes indicate that the distance measures perform significantly worse than the best performing *IC-lin-word-bnc* at the bottom. The solid boxes indicate the corresponding measures are statistically comparable to the *IC-lin-word-bnc* in terms of the mean accuracy they can achieve.

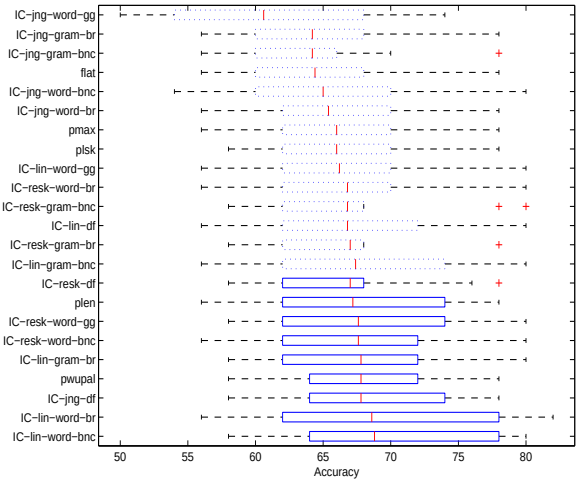


Figure 2: Accuracy on Brown Corpus (10 genres).

Results on 15-genre Brown Corpus. We perform experiments on all 15 genres on the end level of the Brown corpus. The increase of genre classes leads to reduced classification performance. In our experiment, the flat SVM achieves an accuracy of 52.40%, and the structural SVM using path length measure achieves 55.40%, a difference significant at the 5% level. The structural SVMs using information content measures *IC-lin-gram-bnc and IC-resk-word-br* also perform equally well. In addition, we improve on the training accuracy of 52% reported in (Karlgren and Cutting, 1994).

We are also interested in structural accuracy (S-Acc) to see whether the structural SVMs make fewer "big" mistakes. Table 2 shows a cross comparison of structural accuracy. Each row shows how accurate the corresponding method is under the structural accuracy criteria given in the column. The 'no-struct' column corresponds to vanilla accuracy. It is natural to expect each diagonal entry of the numeric table to be the highest, since the respective method is optimised for its own structural distance. However, in our case, Lin's information content measure and the plen measure perform well under any structural accuracy evaluation measure and outperform flat SVMs.

4.6 Other Corpora

In spite of the promising results on the Brown Corpus, structural SVMs on other corpora (BNC, HGC, Syracuse) did not show considerable improvement.

HGC contains 1330 documents divided into 32 approximately equally frequent classes. Its hierarchy has just two levels. Standard accuracy for the best performing structural methods on HGC is just the same as for flat SVM (69.1%), with marginally better structural accuracy (for example, 71.39 vs. 71.04%, using a path-length based structural accuracy). The BNC corpus contains 70 genres and 4053 documents. The number of documents per class ranges from 2 to 501. The accuracy of SSVM is also just comparable to flat SVM (73.6%). The Syracuse corpus is a recently developed large collection of 3027 annotated webpages divided into 292 genres (Crowston et al., 2009). Focusing only on genres containing 15 or more examples, we arrived at a corpus of 2293 samples and 52 genres. Accuracy for flat (53.3%) and structural SVMs (53.7%) are again comparable.

5 Discussion

Given that structural learning can help in topical classification tasks (Tsochantaridis et al., 2005; Dekel et al., 2004), the lack of success on genres is surprising. We now discuss potential reasons for this lack of success.

5.1 Tree Depth and Balance

Our best results were achieved on the Brown corpus, whose genre tree has at least three attractive properties. Firstly, it has a depth greater than 2, i.e. several levels are distinguished. Secondly, it seems visually balanced: branches from root to leaves (or terminals) are of pretty much equal length; branching factors are similar, for example ranging between 2 and 6 for the last level of branching. Thirdly, the number of examples at

Table 2: Structural Accuracy on Brown 15-genre Classification.

Method	no-struct (=typical accuracy)	IC-lin-gram-bnc	plen	IC-resk-word-br	IC-jng-word-gg
flat	52.40	55.34	60.60	58.91	52.19
IC-lin-gram-bnc	55.00	58.15	63.59	61.83	53.85
plen	**55.40**	**58.74**	**64.51**	**62.61**	**54.27**
IC-resk-word-br	55.00	58.24	63.96	62.08	54.08
IC-jng-word-gg	46.00	49.00	54.89	53.01	52.58

each leaf node is roughly comparable (distributional balance).

The other hierarchies violate these properties to a large extent. Thus, the genres in HGC are almost represented by a flat list with just one extra level over 32 categories. Similarly, the vast majority of genres in the Syracuse corpus are also organised in two levels only. Such flat hierarchies do not offer much scope to improve over a completely flat list. There are considerably more levels in the BNC for some branches, e.g., *written/national/broadsheet/arts*, but many other genres are still only specified to the second level of its hierarchy, e.g., *written/adverts*. In addition, the BNC is also distributionally imbalanced, i.e. the number of documents per class varies from 2 to 501 documents.

To test our hypothesis, we tried to skew the Brown genre tree in two ways. First, we kept the tree relatively balanced visually and distributionally but flattened it by removing the second layer *Press, Misc, Non-Fiction, Fiction* from the hierarchy, leaving a tree with only two layers. Second, we skewed the visual and distributional balance of the tree by collapsing its three leaf-level genres under *Press*, and the two under *non-fiction*, leading to 12 genres to classify (cf. Figure 1).

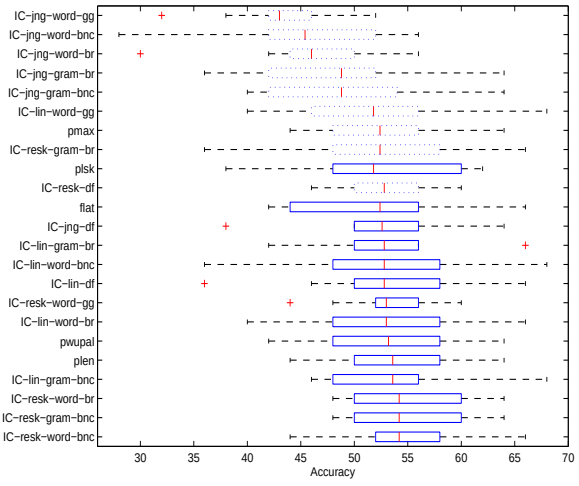


Figure 3: Accuracy on flattened Brown Corpus (15 genres).

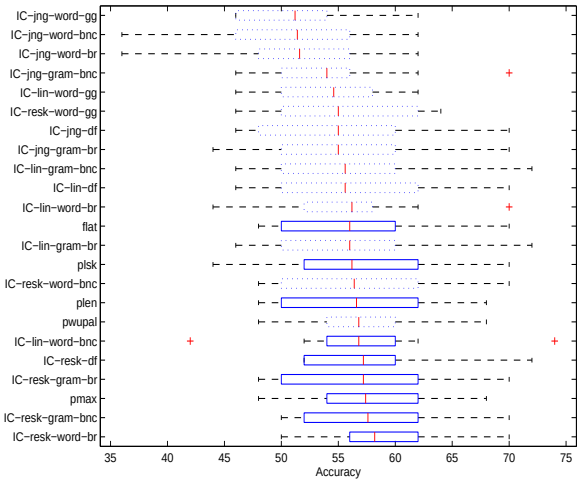


Figure 4: Accuracy on skewed Brown Corpus (12 genres).

As expected, the structural methods on either skewed or flattened hierarchies are not significantly better than the flat SVM. For the flattened hierarchy of 15 leaf genres the maximal accuracy is 54.2% vs. 52.4% for the flat SVM (Figure 3), a non-significant improvement. Similarly, the maximal accuracy on the skewed 12-genre hierarchy is 58.2% vs. 56% (see also Figure 4), again a not significant improvement.

To measure the degree of balance of a tree, we introduce two tree balance scores based on entropy. First, for both measures we extend all branches to the maximum depth of the tree. Then level by level we calculate an entropy score, either according to how many tree nodes at the next level belong to a node at this level (denoted as **vb: visual balance**), or according to how many end level documents belong to a node at this level (denoted as **db: distribution balance**). To make trees with different numbers of internal nodes and leaves more comparable, the entropy score at each level is normalized by the maximal entropy achieved by a tree with uniform distribution of nodes/documents, which is simply $-log(1/N)$, where N denotes the number of nodes at the corre-

sponding level. Finally, the entropy scores for all levels are averaged. It can be shown that any perfect N-ary tree will have the largest visual balance score of 1. If in addition its nodes at each level contain the same number of documents, the distribution balance score will reach the maximum, too.

Table 3 shows the balance scores for all the corpora we use. The first two rows for the Brown corpus have both large visual balance and distribution balance scores. As shown earlier, for those two setups the structural SVMs perform better than the flat approach. In contrast, for the tree hierarchies of Brown that we deformed or flattened, and also BNC and Syracuse, either or both of the two balance scores tend to be lower, and no improvement has been obtained over the flat approach. This may indicate that a further exploration of the relation between tree balance and the performance of structural SVMs is warranted. However, high visual balance and distribution scores do not necessarily imply high performance of structural SVMs, as very flat trees are also visually very balanced. As an example, HGC has a high visual balance score due to a shallow hierarchy and a high distributional balance score due to a roughly equal number of documents contained in each genre. However, HGC did not benefit from structural learning as it is also a very shallow hierarchy; therefore we think that a third variable depth also needs to be taken into account.

A similar observation on the importance of well-balanced hierarchies comes from a recent Pascal challenge on large scale hierarchical text classification,[2] which shows that some flat approaches perform competitively in topic classification with imbalanced hierarchies. However, the participants do not explore explicitly the relation between tree balance and performance.

Other methods for measuring tree balance (some of which are related to ours) are used in the field of phylogenetic research (Shao and Sokal, 1990) but they are only applicable to visual balance. In addition, the methods they used often provide conflicting results on which trees are considered as balanced (Shao and Sokal, 1990).

Table 3: Tree Balance Scores

Corpus	depth	vb	db
Brown (10 genres)	3	**0.9115**	**0.9024**
Brown (15 genres)	3	**0.9186**	**0.9083**
Brown (15, flattened)	2	0.9855	0.8742
Brown (12, skewed)	3	0.8747	0.8947
HGC (32)	2	**0.9562**	**0.9570**
BNC (70)	4	0.9536	0.8039
Syracuse (52)	3	0.9404	0.8634

5.2 Distance Measures

We also scrutinise our distance measures as these are crucial for the structural approach. We notice that simple path length based measures perform well overall; again for the Brown corpus this is probably due to its balanced hierarchy which makes path length appropriate. There are other probable reasons why information content based measures do not perform better than path-length based ones. When measured via document frequency in a corpus we do not have sufficiently large, representative genre-annotated corpora to hand. When measured via genre label frequency, we run into at least two problems. Firstly, as mentioned in Section 3.2.1 genre label frequency does not have to correspond to class frequency of documents. Secondly, the labels used are often abbreviations (e.g. *W_institut_doc, W_newsp_brdsht_nat_social* in BNC Corpus), underspecified (*other, misc, unclassified*) or a collection of phrases (e.g. *belles letters, etc.* in Brown). This made search for frequency very approximate and also loosens the link between label and content.

We investigated in more depth how well the different distance measures are aligned. We adapt the alignment measure between kernels (Cristianini et al., 2002), to investigate how close the distance matrices are. For two distance matrices H_1 and H_2, their alignment $A(H_1, H_2)$ is defined as:

$$\frac{< H_1, H_2 >_F}{\sqrt{< H_1, H_1 >_F, < H_2, H_2 >_F}}, \quad (17)$$

where $< H_1, H_2 >_F = \sum_{i,j}^{k} H_1(g_i, g_j) H_2(g_i, g_j)$ which is the total sum of the entry-wise products between the two distance matrices. Figure 5 shows several distance matrices on the (original) 15 genre Brown corpus. The *plen* matrix has clear blocks for the super genres *press, informative, imaginative*, etc. The *IC-lin-gram-bnc* matrix refines distances in the blocks, due to the introduction of information content. It keeps an alignment score that is over 0.99 (the maximum is 1.00) toward the *plen* matrix, and still has visible block patterns. However, the *IC-jng-word-bnc* significantly adjusts the

[2] http://lshtc.iit.demokritos.gr/

distance entries, has a much lower alignment score with the *plen* matrix, and doesn't reveal apparent blocks. This partially explains the bad performance of the Jiang distance measure on the Brown corpus (see Section 4). The diagrams also show the high closeness between the best performing IC measure and the simple path length based measure.

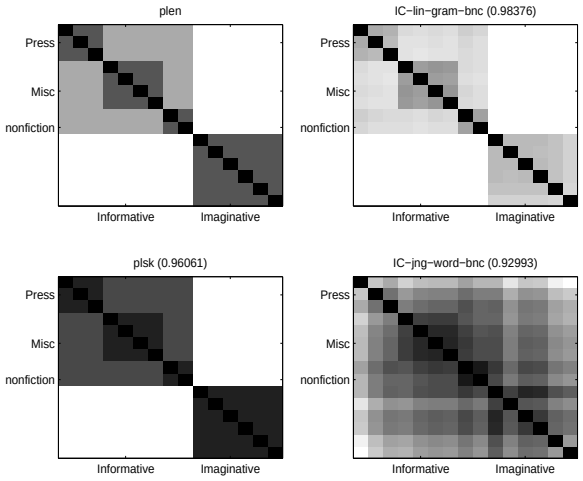


Figure 5: Distance Matrices on Brown. Values in bracket is the alignment with the *plen* matrix

An alternative to structural distance measures would be distance measures between the genres based on pairwise cosine similarities between them. To assess this, we aggregated all character 4-gram training vectors of each genre and calculated standard cosine similarities. Note that these similarities are based on the documents only and do not make use of the Brown hierarchy at all. After converting the similarities to distance, we plug the distance matrix into our structural SVM. However, accuracy on the Brown corpus (15 genres) was almost the same as for a flat SVM. Inspecting the distance matrix visually, we determined that the cosine similarity could clearly distinguish between Fiction and Non-Fiction texts but not between any other genres. This also indicates that the genre structural hierarchy clearly gives information not present in the simple character 4-gram features we use. For a more detailed discussion of the problems of the currently prevalently used character n-grams as features for genre classification, we refer the reader to (Sharoff et al., 2010).

6 Conclusions

In this paper, we have evaluated structural learning approaches to genre classification using several different genre distance measures. Although we were able to improve on non-structural approaches for the Brown corpus, we found it hard to improve over flat SVMs on other corpora. As potential reasons for this negative result, we suggest that current genre hierarchies are either not of sufficient depth or are visually or distributionally imbalanced. We think further investigation into the relationship between hierarchy balance and structural learning is warranted. Further investigation is also needed into the appropriateness of n-gram features for genre identification as well as good measures of genre distance.

In the future, an important task would be the refinement or unsupervised generation of new hierarchies, using information theoretic or data-driven approaches. For a full assessment of hierarchical learning for genre classification, the field of genre studies needs a testbed similar to the Reuters or 20 Newsgroups datasets used in topic-based IR with a balanced genre hierarchy and a representative corpus of reliably annotated webpages.

With regard to algorithms, we are also interested in other formulations for structural SVMs and their large-scale implementation as well as the combination of different distance measures, for example in ensemble learning.

Acknowledgements

We would like to thank the authors of each corpus collection, who invested a lot of effort into producing them. We are also grateful to Google Inc. for supporting this research via their Google Research Awards programme.

References

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA. ACM.

Crammer, K. and Singer, Y. (2002). On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292.

Cristianini, N., Shawe-Taylor, J., and Kandola, J. (2002). On kernel target alignment. In *Proceedings of the Neural Information Process-*

ing Systems, NIPS'01, pages 367–373. MIT Press.

Crowston, K., Kwasnik, B., and Rubleske, J. (2009). Problems in the use-centered development of a taxonomy of web genres. In Mehler, A., Sharoff, S., and Santini, M., editors, *Genres on the Web: Computational Models and Empirical Studies*. Springer, Berlin/New York.

Dekel, O., Keshet, J., and Singer, Y. (2004). Large margin hierarchical classification. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 27, New York, NY, USA. ACM.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923.

Giesbrecht, E. and Evert, S. (2009). Part-of-Speech (POS) Tagging - a solved task? An evaluation of POS taggers for the Web as corpus. In *Proceedings of the Fifth Web as Corpus Workshop (WAC5)*, pages 27–35, Donostia-San Sebastián.

Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *CoRR*, cmp-lg/9709008.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods – Support Vector Learning*, pages 41–56. MIT Press.

Joachims, T., Finley, T., and Yu, C.-N. (2009). Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Kanaris, I. and Stamatatos, E. (2009). Learning to recognize webpage genres. *Information Processing and Management*, 45:499–512.

Karlgren, J. and Cutting, D. (1994). Recognizing text genres with simple metrics using discriminant analysis. In *Proc. of the 15th. International Conference on Computational Linguistics (*COLING 94*)*, pages 1071 – 1075, Kyoto, Japan.

Keerthi, S. S., Sundararajan, S., Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2008). A sequential dual method for large scale multi-class linear svms. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 408–416, New York, NY, USA. ACM.

Kessler, B., Nunberg, G., and Schütze, H. (1997). Automatic detection of text genre. In *Proceedings of the 35^{th} ACL/8^{th} EACL*, pages 32–38.

Kučera, H. and Francis, W. N. (1967). *Computational analysis of present-day American English*. Brown University Press, Providence.

Leacock, C. and Chodorow, M. (1998). *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press.

Lee, D. (2001). Genres, registers, text types, domains, and styles: clarifying the concepts and navigating a path through the BNC jungle. *Language Learning and Technology*, 5(3):37–72.

Lin, D. (1998). An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Meyer zu Eissen, S. and Stein, B. (2004). Genre classification of web pages. In *Proceedings of the 27th German Conference on Artificial Intelligence*, Ulm, Germany.

Pedersen, T., Pakhomov, S. V. S., Patwardhan, S., and Chute, C. G. (2007). Measures of semantic similarity and relatedness in the biomedical domain. *J. of Biomedical Informatics*, 40(3):288–299.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Santini, M. (2007). *Automatic Identification of Genre in Web Pages*. PhD thesis, University of Brighton.

Shao, K.-T. and Sokal, R. R. (1990). Tree balance. *Systematic Zoology*, 39(3):266–276.

Sharoff, S., Wu, Z., and Markert, K. (2010). The Web library of Babel: evaluating genre collections. In *Proc. of the Seventh Language Resources and Evaluation Conference, LREC 2010*, Malta.

Stubbe, A. and Ringlstetter, C. (2007). Recognizing genres. In Santini, M. and Sharoff, S., editors, *Proc. Towards a Reference Corpus of Web Genres*.

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484.

Vidulin, V., Luštrek, M., and Gams, M. (2007). Using genres to improve search engines. In *Proc. Towards Genre-Enabled Search Engines: The Impact of NLP. RANLP-07*.

Webber, B. (2009). Genre distinctions for discourse in the Penn TreeBank. In *Proc the 47th Annual Meeting of the ACL*, pages 674–682.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.

Metadata-Aware Measures for Answer Summarization in Community Question Answering

Mattia Tomasoni *
Dept. of Information Technology
Uppsala University, Uppsala, Sweden
mattia.tomasoni.8371@student.uu.se

Minlie Huang
Dept. Computer Science and Technology
Tsinghua University, Beijing 100084, China
aihuang@tsinghua.edu.cn

Abstract

This paper presents a framework for automatically processing information coming from community Question Answering (cQA) portals with the purpose of generating a trustful, complete, relevant and succinct summary in response to a question. We exploit the metadata intrinsically present in User Generated Content (UGC) to bias automatic multi-document summarization techniques toward high quality information. We adopt a representation of concepts alternative to n-grams and propose two concept-scoring functions based on semantic overlap. Experimental results on data drawn from Yahoo! Answers demonstrate the effectiveness of our method in terms of ROUGE scores. We show that the information contained in the best answers voted by users of cQA portals can be successfully complemented by our method.

1 Introduction

Community Question Answering (cQA) portals are an example of Social Media where the information need of a user is expressed in the form of a question for which a best answer is picked among the ones generated by other users. cQA websites are becoming an increasingly popular complement to search engines: overnight, a user can expect a human-crafted, natural language answer tailored to her specific needs. We have to be aware, though, that User Generated Content (UGC) is often redundant, noisy and untrustworthy (Jeon et al., 2006; Wang et al., 2009b; Suryanto et al., 2009). Interestingly, a great amount of information is embedded in the metadata generated as a byproduct of users' action and interaction on Social Media. Much valuable information is contained in answers other than the chosen best one (Liu et al., 2008). Our work aims to show that such information can be successfully extracted and made available by exploiting metadata to distill cQA content. To this end, we casted the problem to an instance of the query-biased multi-document summarization task, where the question was seen as a query and the available answers as documents to be summarized. We mapped each characteristic that an ideal answer should present to a measurable property that we wished the final summary could exhibit:

- Quality to assess trustfulness in the source,
- Coverage to ensure completeness of the information presented,
- Relevance to keep focused on the user's information need and
- Novelty to avoid redundancy.

Quality of the information was assessed via Machine Learning (ML) techniques under best answer supervision in a vector space consisting of linguistic and statistical features about the answers and their authors. Coverage was estimated by semantic comparison with the knowledge space of a corpus of answers to similar questions which had been retrieved through the Yahoo! Answers API [1]. Relevance was computed as information overlap between an answer and its question, while Novelty was calculated as inverse overlap with all other answers to the same question. A score was assigned to each concept in an answer according to

The research was conducted while the first author was visiting Tsinghua University.

[1] http://developer.yahoo.com/answers

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 760–769,
Uppsala, Sweden, 11-16 July 2010.

the above properties. A score-maximizing summary under a maximum coverage model was then computed by solving an associated Integer Linear Programming problem (Gillick and Favre, 2009; McDonald, 2007). We chose to express concepts in the form of Basic Elements (BE), a semantic unit developed at ISI[2] and modeled semantic overlap as intersection in the equivalence classes of two concepts (formal definitions will be given in section 2.3).

The objective of our work was to present what we believe is a valuable conceptual framework; more advance machine learning and summarization techniques would most likely improve the performances.

The remaining of this paper is organized as follows. In the next section Quality, Coverage, Relevance and Novelty measures are presented; we explain how they were calculated and combined to generate a final summary of all answers to a question. Experiments are illustrated in Section 3, where we give evidence of the effectiveness of our method. We list related work in Section 5, discuss possible alternative approaches in Section 4 and provide our conclusions in Section 6.

2 The summarization framework

2.1 Quality as a ranking problem

Quality assessing of information available on Social Media had been studied before mainly as a binary classification problem with the objective of detecting low quality content. We, on the other hand, treated it as a ranking problem and made use of quality estimates with the novel intent of successfully combining information from sources with different levels of trustfulness and writing ability. This is crucial when manipulating UGC, which is known to be subject to particularly great variance in credibility (Jeon et al., 2006; Wang et al., 2009b; Suryanto et al., 2009) and may be poorly written.

An answer a was given along with information about the user u that authored it, the set TA^q (Total Answers) of all answers to the same question q and the set TA^u of all answers by the same user. Making use of results available in the literature (Agichtein et al., 2008) [3], we designed a Quality feature space to capture the following syntactic, behavioral and statistical properties:

- ϑ, length of answer a
- ς, number of non-stopwords in a with a corpus frequency larger than n (set to 5 in our experiments)
- ϖ, points awarded to user u according to the Yahoo! Answers' points system
- ϱ, ratio of best answers posted by user u

The features mentioned above determined a space Ψ; An answer a, in such feature space, assumed the vectorial form:

$$\Psi^a = (\ \vartheta,\ \varsigma,\ \varpi,\ \varrho\ )$$

Following the intuition that chosen best answers ($a^\star$) carry high quality information, we used supervised ML techniques to predict the probability of a to have been selected as a best answer $a^\star$. We trained a Linear Regression classifier to learn the weight vector $W = (w_1, w_2, w_3, w_4)$ that would combine the above feature. Supervision was given in the form of a training set Tr^Q of labeled pairs defined as:

$$Tr^Q = \{\langle\ \Psi^a,\ isbest^a\ \rangle\}$$

$isbest^a$ was a boolean label indicating whether a was an $a^\star$ answer; the training set size was determined experimentally and will be discussed in Section 3.2. Although the value of $isbest^a$ was known for all answers, the output of the classifier offered us a real-valued prediction that could be interpreted as a quality score $Q(\Psi^a)$:

$$\begin{aligned} Q(\Psi^a) &\approx P(\ isbest^a = 1 \mid a, u, TA^u,\ ) \\ &\approx P(\ isbest^a = 1 \mid \Psi^a\ ) \\ &= W^T \cdot \Psi^a \end{aligned} \quad (1)$$

The Quality measure for an answer a was approximated by the probability of such answer to be a best answer ($isbest^a = 1$) with respect to its author u and the sets TA^u and TA^q. It was calculated as dot product between the learned weight vector W and the feature vector for answer Ψ^a.

Our decision to proceed in an unsupervised direction came from the consideration that any use of external human annotation would have made it impracticable to build an actual system on larger scale. An alternative, completely unsupervised approach to quality detection that has not undergone experimental analysis is discussed in Section 4.

[2] Information Sciences Institute, University of Southern California, http://www.isi.edu

[3] A long list of features is proposed; training a classifier on all of them would no doubt increase the performances.

2.2 Bag-of-BEs and semantic overlap

The properties that remain to be discussed, namely Coverage, Relevance and Novelty, are measures of semantic overlap between concepts; a concept is the smallest unit of meaning in a portion of written text. To represent sentences and answers we adopted an alternative approach to classical n-grams that could be defined bag-of-BEs. a BE is "a head|modifier|relation triple representation of a document developed at ISI" (Zhou et al., 2006). BEs are a strong theoretical instrument to tackle the ambiguity inherent in natural language that find successful practical applications in real-world query-based summarization systems. Different from n-grams, they are variant in length and depend on parsing techniques, named entity detection, part-of-speech tagging and resolution of syntactic forms such as hyponyms, pronouns, pertainyms, abbreviation and synonyms. To each BE is associated a class of semantically equivalent BEs as result of what is called a transformation of the original BE; the mentioned class uniquely defines the concept. What seemed to us most remarkable is that this makes the concept context-dependent. A sentence is defined as a set of concepts and an answer is defined as the union between the sets that represent its sentences.

The rest of this section gives formal definition of our model of concept representation and semantic overlap. From a set-theoretical point of view, each concepts c was uniquely associated with a set $E^c = \{c_1, c_2 \dots c_m\}$ such that:

$$\forall i, j \quad (c_i \approx^L c) \wedge (c_i \not\equiv c) \wedge (c_i \not\equiv c_j)$$

In our model, the "$\equiv$" relation indicated syntactic equivalence (exact pattern matching), while the "$\approx^L$" relation represented semantic equivalence under the convention of some language L (two concepts having the same meaning). E^c was defined as the set of semantically equivalent concepts to c, called its equivalence class; each concept c_i in E^c carried the same meaning ($\approx^L$) of concept c without being syntactically identical ($\equiv$); furthermore, no two concepts i and j in the same equivalence class were identical.

"Climbing a tree to escape a black bear is pointless because *they can climb* very well."

BE = they|climb
E^c = {climb|bears, bear|go_up, climbing|animals, climber|instincts, trees|go_up, claws|climb...}

Given two concepts c and k:

$$c \bowtie k \begin{cases} c \equiv k & \text{or} \\ E^c \cap E^k \neq \emptyset \end{cases}$$

We defined semantic overlap as occurring between c and k if they were syntactically identical or if their equivalence classes E^c and E^k had at least one element in common. In fact, given the above definition of equivalence class and the transitivity of "$\equiv$" relation, we have that if the equivalence classes of two concepts are not disjoint, then they must bare the same meaning under the convention of some language L; in that case we said that c semantically overlapped k. It is worth noting that relation "$\bowtie$" is symmetric, transitive and reflexive; as a consequence all concepts with the same meaning are part of a same equivalence class. BE and equivalence class extraction were performed by modifying the behavior of the BEwT-E-0.3 framework [4]. The framework itself is responsible for the operative definition of the "$\approx^L$" relation and the creation of the equivalence classes.

2.3 Coverage via concept importance

In the scenario we proposed, the user's information need is addressed in the form of a unique, summarized answer; information that is left out of the final summary will simply be unavailable. This raises the concern of completeness: besides ensuring that the information provided could be trusted, we wanted to guarantee that the posed question was being answered thoroughly. We adopted the general definition of Coverage as the portion of relevant information about a certain subject that is contained in a document (Swaminathan et al., 2009). We proceeded by treating each answer to a question q as a separate document and we retrieved through the Yahoo! Answers API a set TK^q (Total Knowledge) of 50 answers [5] to questions similar to q: the knowledge space of TK^q was chosen to approximate the entire knowledge space related to the queried question q. We calculated Coverage as a function of the portion of answers in TK^q that presented semantic overlap with a.

[4] The authors can be contacted regarding the possibility of sharing the code of the modified version. Original version available from `http://www.isi.edu/publications/licensed-sw/BE/index.html`.

[5] such limit was imposed by the current version of the API. Experiments with a greater corpus should be carried out in the future.

$$C(a,q) = \sum_{c_i \in a} \gamma(c_i) \cdot tf(c_i, a) \quad (2)$$

The Coverage measure for an answer a was calculated as the sum of term frequency $tf(c_i, a)$ for concepts in the answer itself, weighted by a concept importance function, $\gamma(c_i)$, for concepts in the total knowledge space TK^q. $\gamma(c)$ was defined as follows:

$$\gamma(c) = \frac{|TK^{q,c}|}{|TK^q|} \cdot \log_2 \frac{|TK^q|}{|TK^{q,c}|} \quad (3)$$

where $TK^{q,c} = \{d \in TK^q : \exists k \in d,\ k \bowtie c\}$

The function $\gamma(c)$ of concept c was calculated as a function of the cardinality of set TK^q and set $TK^{q,c}$, which was the subset of all those answers d that contained at least one concept k which presented semantical overlap with c itself. A similar idea of knowledge space coverage is addressed by Swaminathan et al. (2009), from which formulas (2) and (3) were derived.

A sensible alternative would be to estimate Coverage at the sentence level.

2.4 Relevance and Novelty via $\bowtie$ relation

To this point, we have addressed matters of trustfulness and completeness. Another widely shared concern for Information Retrieval systems is Relevance to the query. We calculated relevance by computing the semantic overlap between concepts in the answers and the question. Intuitively, we reward concepts that express meaning that could be found in the question to be answered.

$$R(c,q) = \frac{|q^c|}{|q|} \quad (4)$$

where $q^c = \{k \in q : k \bowtie c\}$

The Relevance measure $R(c,q)$ of a concept c with respect to a question q was calculated as the ratio of the cardinality of set q^c (containing all concepts in q that semantically overlapped with c) normalized by the total number of concepts in q.

Another property we found desirable, was to minimize redundancy of information in the final summary. Since all elements in TA^q (the set of all answers to q) would be used for the final summary, we positively rewarded concepts that were expressing novel meanings.

$$N(c,q) = 1 - \frac{|TA^{q,c}|}{|TA^q|} \quad (5)$$

where $TA^{q,c} = \{d \in TA^q : \exists k \in d,\ k \bowtie c\}$

The Novelty measure $N(c,q)$ of a concept c with respect to a question q was calculated as the ratio of the cardinality of set $TA^{q,c}$ over the cardinality of set TA^q; $TA^{q,c}$ was the subset of all those answers d in TA^q that contained at least one concept k which presented semantical overlap with c.

2.5 The concept scoring functions

We have now determined how to calculate the scores for each property in formulas (1), (2), (4) and (5); under the assumption that the Quality and Coverage of a concept are the same of its answer, every concept c part of an answer a to some question q, could be assigned a score vector as follows:

$$\Phi^c = (\ Q(\Psi^a),\ C(a,q),\ R(c,q),\ N(c,q)\ )$$

What we needed at this point was a function S of the above vector which would assign a higher score to concepts most worthy of being included in the final summary. Our intuition was that since Quality, Coverage, Novelty and Relevance were all virtues properties, S needed to be monotonically increasing with respect to all its dimensions. We designed two such functions. Function (6), which multiplied the scores, was based on the probabilistic interpretation of each score as an independent event. Further empirical considerations, brought us to later introduce a logarithmic component that would discourage inclusion of sentences shorter then a threshold t (a reasonable choice for this parameter is a value around 20). The score for concept c appearing in sentence s^c was calculated as:

$$S^{\Pi}(c) = \prod_{i=1}^{4} (\Phi_i^c) \cdot \log_t(length(s^c)) \quad (6)$$

A second approach that made use of human annotation to learn a vector of weights $V = (v_1, v_2, v_3, v_4)$ that linearly combined the scores was investigated. Analogously to what had been done with scoring function (6), the Φ space was augmented with a dimension representing the length of the answer.

$$S^{\Sigma}(c) = \sum_{i=1}^{4} (\Phi_i^c \cdot v_i) + length(s^c) \cdot v_5 \quad (7)$$

In order to learn the weight vector V that would combine the above scores, we asked three human annotators to generate question-biased extractive summaries based on all answers available for a certain question. We trained a Linear Regression

classifier with a set Tr^S of labeled pairs defined as:

$$Tr^S = \{\langle\, (\Phi^c,\ length(s^c)),\ include^c \,\rangle\}$$

$include^c$ was a boolean label that indicated whether s^c, the sentence containing c, had been included in the human-generated summary; $length(s^c)$ indicated the length of sentence s^c. Questions and relative answers for the generation of human summaries were taken from the "filtered dataset" described in Section 3.1.

The concept score for the same BE in two separate answers is very likely to be different because it belongs to answers with their own Quality and Coverage values: this only makes the scoring function context-dependent and does not interfere with the calculation the Coverage, Relevance and Novelty measures, which are based on information overlap and will regard two BEs with overlapping equivalence classes as being the same, regardless of their score being different.

2.6 Quality constrained summarization

The previous sections showed how we quantitatively determined which concepts were more worthy of becoming part of the final machine summary M. The final step was to generate the summary itself by automatically selecting sentences under a length constraint. Choosing this constraint carefully demonstrated to be of crucial importance during the experimental phase. We again opted for a metadata-driven approach and designed the length constraint as a function of the lengths of all answers to q (TA^q) weighted by the respective Quality measures:

$$length^M = \sum_{a \in TA^q} length(a) \cdot Q(\Psi^a) \quad (8)$$

The intuition was that the longer and the more trustworthy answers to a question were, the more space was reasonable to allocate for information in the final, machine summarized answer M.

M was generated so as to maximize the scores of the concepts it included. This was done under a maximum coverage model by solving the following Integer Linear Programming problem:

$$\text{maximize:} \quad \sum_i S(c_i) \cdot x_i \quad (9)$$

$$\text{subject to:} \quad \sum_j length(j) \cdot s_j \leq length^M$$

$$\sum_j y_j \cdot occ_{ij} \geq x_i \quad \forall i \quad (10)$$

$$occ_{ij},\ x_i,\ y_j \in \{0,1\} \quad \forall i,j$$

$$occ_{ij} = 1 \text{ if } c_i \in s_j, \quad \forall i,j$$

$$x_i = 1 \text{ if } c_i \in M, \quad \forall i$$

$$y_j = 1 \text{ if } s_j \in M, \quad \forall j$$

In the above program, M is the set of selected sentences: $M = \{s_j : y_j = 1, \quad \forall j\}$. The integer variables x_i and y_j were equals to one if the corresponding concept c_i and sentence s_j were included in M. Similarly occ_{ij} was equal to one if concept c_i was contained in sentence s_j. We maximized the sum of scores $S(c_i)$ (for S equals to S^Π or S^Σ) for each concept c_i in the final summary M. We did so under the constraint that the total length of all sentences s_j included in M must be less than the total expected length of the summary itself. In addition, we imposed a consistency constraint: if a concept c_i was included in M, then at least one sentence s_j that contained the concept must also be selected (constraint (10)). The described optimization problem was solved using lp_solve [6].

We conclude with an empirical side note: since solving the above can be computationally very demanding for large number of concepts, we found performance-wise very fruitful to skim about one fourth of the concepts with lowest scores.

3 Experiments

3.1 Datasets and filters

The initial dataset was composed of 216,563 questions and 1,982,006 answers written by 171,676 user in 100 categories from the Yahoo! Answers portal[7]. We will refer to this dataset as the "unfiltered version". The metadata described in section 2.1 was extracted and normalized; quality experiments (Section 3.2) were then conducted. The unfiltered version was later reduced to 89,814 question-answer pairs that showed statistical and linguistic properties which made them particularly adequate for our purpose. In particular, trivial, factoid and encyclopedia-answerable questions were

[6] the version used was lp_solve 5.5, available at `http://lpsolve.sourceforge.net/5.5`

[7] The reader is encouraged to contact the authors regarding the availability of data and filters described in this Section.

removed by applying a series of patterns for the identification of complex questions. The work by Liu et al. (2008) indicates some categories of questions that are particularly suitable for summarization, but due to the lack of high-performing question classifiers we resorted to human-crafted question patterns. Some pattern examples are the following:

- {Why,What is the reason} [...]
- How {to,do,does,did} [...]
- How {is,are,were,was,will} [...]
- How {could,can,would,should} [...]

We also removed questions that showed statistical values outside of convenient ranges: the number of answers, length of the longest answer and length of the sum of all answers (both absolute and normalized) were taken in consideration. In particular we discarded questions with the following characteristics:

- there were less than three answers [8]
- the longest answer was over 400 words (likely a copy-and-paste)
- the sum of the length of all answers outside of the $(100,\ 1000)$ words interval
- the average length of answers was outside of the $(50,\ 300)$ words interval

At this point a second version of the dataset was created to evaluate the summarization performance under scoring function (6) and (7); it was generated by manually selecting questions that arouse subjective, human interest from the previous 89,814 question-answer pairs. The dataset size was thus reduced to 358 answers to 100 questions that were manually summarized (refer to Section 3.3). From now on we will refer to this second version of the dataset as the "filtered version".

3.2 Quality assessing

In Section 2.1 we claimed to be able to identify high quality content. To demonstrate it, we conducted a set of experiments on the original unfiltered dataset to establish whether the feature space Ψ was powerful enough to capture the quality of answers; our specific objective was to estimate the amount of training examples needed to successfully train a classifier for the quality assessing task. The Linear Regression[9] method was chosen to determine the probability $Q(\Psi^a)$ of a to be a best answer to q; as explained in Section 2.1, those probabilities were interpreted as quality estimates. The evaluation of the classifier's output was based on the observation that given the set of all answers TA^q relative to q and the best answer $a^\star$, a successfully trained classifier should be able to rank $a^\star$ ahead of all other answers to the same question. More precisely, we defined Precision as follows:

$$\frac{|\{q \in Tr^Q :\ \forall a \in TA^q,\ Q(\Psi^{a^\star}) > Q(\Psi^a)\}|}{|Tr^Q|}$$

where the numerator was the number of questions for which the classifier was able to correctly rank $a^\star$ by giving it the highest quality estimate in TA^q and the denominator was the total number of examples in the training set Tr^Q. Figure 1 shows the precision values (Y-axis) in identifying best answers as the size of Tr^Q increases (X-axis). The experiment started from a training set of size 100 and was repeated adding 300 examples at a time until precision started decreasing. With each increase in training set size, the experiment was repeated ten times and average precision values were calculated. In all runs, training examples were picked randomly from the unfiltered dataset described in Section 3.1; for details on Tr^Q see Section 2.1. A training set of 12,000 examples was chosen for the summarization experiments.

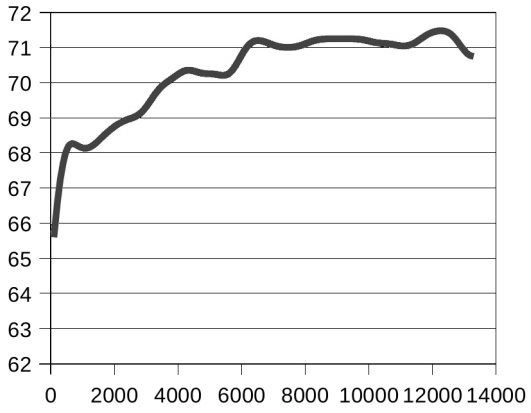


Figure 1: Precision values (Y-axis) in detecting best answers $a^\star$ with increasing training set size (X-axis) for a Linear Regression classifier on the unfiltered dataset.

[8] Being too easy to summarize or not requiring any summarization at all, those questions wouldn't constitute an valuable test of the system's ability to extract information.

[9] Performed with Weka 3.7.0 available at `http://www.cs.waikato.ac.nz/~ml/weka`

System	$a^\star$ (baseline)	S^Σ	S^Π
ROUGE-1_R	51.7%	67.3%	67.4%
ROUGE-1_P	62.2%	54.0%	71.2%
ROUGE-1_F	52.9%	59.3%	66.1%
ROUGE-2_R	40.5%	52.2%	58.8%
ROUGE-2_P	49.0%	41.4%	63.1%
ROUGE-2_F	41.6%	45.9%	57.9%
ROUGE-L_R	50.3%	65.1%	66.3%
ROUGE-L_P	60.5%	52.3%	70.7%
ROUGE-L_F	51.5%	57.3%	65.1%

Table 1: Summarization Evaluation on filtered dataset (refer to Section 3.1 for details). ROUGE-L, ROUGE-1 and ROUGE-2 are presented; for each, Recall (R), Precision (P) and F-1 score (F) are given.

3.3 Evaluating answer summaries

The objective of our work was to summarize answers from cQA portals. Two systems were designed: Table 1 shows the performances using function S^Σ (see equation (7)), and function S^Π (see equation (6)). The chosen best answer $a^\star$ was used as a baseline. We calculated ROUGE-1 and ROUGE-2 scores[10] against human annotation on the filtered version of the dataset presented in Section 3.1. The filtered dataset consisted of 358 answers to 100 questions. For each questions q, three annotators were asked to produce an extractive summary of the information contained in TA^q by selecting sentences subject to a fixed length limit of 250 words. The annotation resulted in 300 summaries (larger-scale annotation is still ongoing). For the S^Σ system, 200 of the 300 generated summaries were used for training and the remaining were used for testing (see the definition of Tr^S Section 2.5). Cross-validation was conducted. For the S^Π system, which required no training, all of the 300 summaries were used as the test set.

S^Σ outperformed the baseline in Recall (R) but not in Precision (P); nevertheless, the combined F-1 score (F) was sensibly higher (around 5 points percentile). On the other hand, our S^Π system showed very consistent improvements of an order of 10 to 15 points percentile over the baseline on all measures; we would like to draw attention on the fact that even if Precision scores are higher, it is on Recall scores that greater improvements were achieved. This, together with the results obtained by S^Σ, suggest performances could benefit from the enforcement of a more stringent length constraint than the one proposed in (8). Further potential improvements on S^Σ could be obtained by choosing a classifier able to learn a more expressive underlying function.

[10]Available at `http://berouge.com/default.aspx`

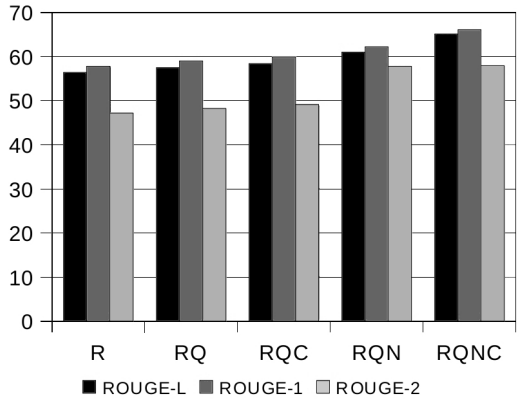


Figure 2: Increase in ROUGE-L, ROUGE-1 and ROUGE-2 performances of the S^Π system as more measures are taken in consideration in the scoring function, starting from Relevance alone (R) to the complete system (RQNC). F-1 scores are given.

In order to determine what influence the single measures had on the overall performance, we conducted a final experiment on the filtered dataset to evaluate (the S^Π scoring function was used). The evaluation was conducted in terms of F-1 scores of ROUGE-L, ROUGE-1 and ROUGE-2. First only Relevance was tested (R) and subsequently Quality was added (RQ); then, in turn, Coverage (RQC) and Novelty (RQN); Finally the complete system taking all measures in consideration (RQNC). Results are shown in Figure 2. In general performances increase smoothly with the exception of ROUGE-2 score, which seems to be particularly sensitive to Novelty: no matter what combination of measures is used (R alone, RQ, RQC), changes in ROUGE-2 score remain under one point percentile. Once Novelty is added, performances rise abruptly to the system's highest. A summary example, along with the question and the best answer, is presented in Table 2.

4 Discussion and Future Directions

We conclude by discussing a few alternatives to the approaches we presented. The $length^M$ constraint for the final summary (Section 2.6), could have been determined by making use of external knowledge such as TK^q: since TK^q represents

HOW TO PROTECT YOURSELF FROM A BEAR? http://answers.yahoo.com/question/index?qid=20060818062414AA7VldB
BEST ANSWER Great question. I have done alot of trekking through California, Montana and Wyoming and have met Black bears (which are quite dinky and placid but can go nuts if they have babies), and have been half an hour away from (allegedly) the mother of all grizzley s whilst on a trail through Glacier National park - so some other trekkerers told me... What the park wardens say is SING, SHOUT, MAKE NOISE...do it loudly, let them know you are there..they will get out of the way, it is a surprised bear wot will go mental and rip your little legs off..No fun permission: anything that will confuse them and stop them in their tracks...I have been told be an native american buddy that to keep a bottle of perfume in your pocket...throw it at the ground near your feet and make the place stink: they have good noses, them bears, and a mega concentrated dose of Britney Spears Obsessive Compulsive is gonna give em something to think about...Have you got a rape alarm? Def take that...you only need to distract them for a second then they will lose interest..Stick to the trails is the most important thing, and talk to everyone you see when trekking: make sure others know where you are.
SUMMARIZED ANSWER […] In addition if the bear actually approaches you or charges you.. still stand your ground. Many times they will not actually come in contact with you, they will charge, almost touch you than run away. […] The actions you should take are different based on the type of bear. for example adult Grizzlies can t climb trees, but Black bears can even when adults. They can not climb in general as thier claws are longer and not semi-retractable like a Black bears claws. […] I truly disagree with the whole play dead approach because both Grizzlies and Black bears are oppurtunistic animals and will feed on carrion as well as kill and eat animals. Although Black bears are much more scavenger like and tend not to kill to eat as much as they just look around for scraps. Grizzlies on the other hand are very accomplished hunters and will take down large prey animals when they want. […] I have lived in the wilderness of Northern Canada for many years and I can honestly say that Black bears are not at all likely to attack you in most cases they run away as soon as they see or smell a human, the only places where Black bears are agressive is in parks with visitors that feed them, everywhere else the bears know that usually humans shoot them and so fear us. […]

Table 2: A summarized answer composed of five different portions of text generated with the S^{Π} scoring function; the chosen best answer is presented for comparison. The richness of the content and the good level of readability make it a successful instance of metadata-aware summarization of information in cQA systems. Less satisfying examples include summaries to questions that require a specific order of sentences or a compromise between strongly discordant opinions; in those cases, the summarized answer might lack logical consistency.

the total knowledge available about q, a coverage estimate of the final answers against it would have been ideal. Unfortunately the lack of metadata about those answers prevented us from proceeding in that direction. This consideration suggests the idea of building TK^q using similar answers in the dataset itself, for which metadata is indeed available. Furthermore, similar questions in the dataset could have been used to augment the set of answers used to generate the final summary with answers coming from similar questions. Wang et al. (2009a) presents a method to retrieve similar questions that could be worth taking in consideration for the task. We suggest that the retrieval method could be made Quality-aware. A Quality feature space for questions is presented by Agichtein et al. (2008) and could be used to rank the quality of questions in a way similar to how we ranked the quality of answers.

The Quality assessing component itself could be built as a module that can be adjusted to the kind of Social Media in use; the creation of customized Quality feature spaces would make it possible to handle different sources of UGC (forums, collaborative authoring websites such as Wikipedia, blogs etc.). A great obstacle is the lack of systematically available high quality training examples: a tentative solution could be to make use of clustering algorithms in the feature space; high and low quality clusters could then be labeled by comparison with examples of virtuous behavior (such as Wikipedia's Featured Articles). The quality of a document could then be estimated as a function of distance from the centroid of the cluster it belongs to. More careful estimates could take the position of other clusters and the concentration of nearby documents in consideration.

Finally, in addition to the chosen best answer, a DUC-styled query-focused multi-document summary could be used as a baseline against which the performances of the system can be checked.

5 Related Work

A work with a similar objective to our own is that of Liu et al. (2008), where standard multi-document summarization techniques are employed along with taxonomic information about questions. Our approach differs in two fundamental aspects: it took in consideration the peculiarities of the data in input by exploiting the nature of UGC and available metadata; additionally, along with relevance, we addressed challenges that are specific to Question Answering, such as Coverage and Novelty. For an investigation of Coverage in the context of Search Engines, refer to Swaminathan et al. (2009).

At the core of our work laid information trustfulness, summarization techniques and alternative concept representation. A general approach to the broad problem of evaluating information credibility on the Internet is presented by Akamine et al. (2009) with a system that makes use of semantic-aware Natural Language Preprocessing techniques. With analogous goals, but a focus on UGC, are the papers of Stvilia et al. (2005), Mcguinness et al. (2006), Hu et al. (2007) and

Zeng et al. (2006), which present a thorough investigation of Quality and trust in Wikipedia. In the cQA domain, Jeon et al. (2006) presents a framework to use Maximum Entropy for answer quality estimation through non-textual features; with the same purpose, more recent methods based on the expertise of answerers are proposed by Suryanto et al. (2009), while Wang et al. (2009b) introduce the idea of ranking answers taking their relation to questions in consideration. The paper that we regard as most authoritative on the matter is the work by Agichtein et al. (2008) which inspired us in the design of the Quality feature space presented in Section 2.1.

Our approach merged trustfulness estimation and summarization techniques: we adapted the automatic concept-level model presented by Gillick and Favre (2009) to our needs; related work in multi-document summarization has been carried out by Wang et al. (2008) and McDonald (2007). A relevant selection of approaches that instead make use of ML techniques for query-biased summarization is the following: Wang et al. (2007), Metzler and Kanungo (2008) and Li et al. (2009). An aspect worth investigating is the use of partially labeled or totally unlabeled data for summarization in the work of Wong et al. (2008) and Amini and Gallinari (2002).

Our final contribution was to explore the use of Basic Elements document representation instead of the widely used n-gram paradigm: in this regard, we suggest the paper by Zhou et al. (2006).

6 Conclusions

We presented a framework to generate trustful, complete, relevant and succinct answers to questions posted by users in cQA portals. We made use of intrinsically available metadata along with concept-level multi-document summarization techniques. Furthermore, we proposed an original use for the BE representation of concepts and tested two concept-scoring functions to combine Quality, Coverage, Relevance and Novelty measures. Evaluation results on human annotated data showed that our summarized answers constitute a solid complement to best answers voted by the cQA users.

We are in the process of building a system that performs on-line summarization of large sets of questions and answers from Yahoo! Answers. Larger-scale evaluation of results against other state-of-the-art summarization systems is ongoing.

Acknowledgments

This work was partly supported by the Chinese Natural Science Foundation under grant No. 60803075, and was carried out with the aid of a grant from the International Development Research Center, Ottawa, Canada. We would like to thank Prof. Xiaoyan Zhu, Mr. Yang Tang and Mr. Guillermo Rodriguez for the valuable discussions and comments and for their support. We would also like to thank Dr. Chin-yew Lin and Dr. Eugene Agichtein from Emory University for sharing their data.

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In Marc Najork, Andrei Z. Broder, and Soumen Chakrabarti, editors, *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*, pages 183–194. ACM.

Susumu Akamine, Daisuke Kawahara, Yoshikiyo Kato, Tetsuji Nakagawa, Kentaro Inui, Sadao Kurohashi, and Yutaka Kidawara. 2009. Wisdom: a web information credibility analysis system. In *ACL-IJCNLP '09: Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 1–4, Morristown, NJ, USA. Association for Computational Linguistics.

Massih-Reza Amini and Patrick Gallinari. 2002. The use of unlabeled data to improve supervised learning for text summarization. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 105–112, New York, NY, USA. ACM.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *ILP '09: Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18, Morristown, NJ, USA. Association for Computational Linguistics.

Meiqun Hu, Ee-Peng Lim, Aixin Sun, Hady Wirawan Lauw, and Ba-Quy Vuong. 2007. Measuring article quality in wikipedia: models and evaluation. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 243–252, New York, NY, USA. ACM.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of

answers with non-textual features. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 228–235, New York, NY, USA. ACM.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 71–80, New York, NY, USA. ACM.

Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. 2008. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 497–504, Manchester, UK, August. Coling 2008 Organizing Committee.

Ryan T. McDonald. 2007. A study of global inference algorithms in multi-document summarization. In Giambattista Amati, Claudio Carpineto, and Giovanni Romano, editors, *ECIR*, volume 4425 of *Lecture Notes in Computer Science*, pages 557–564. Springer.

Deborah L. Mcguinness, Honglei Zeng, Paulo Pinheiro Da Silva, Li Ding, Dhyanesh Narayanan, and Mayukh Bhaowal. 2006. Investigation into trust for collaborative information repositories: A wikipedia case study. In *In Proceedings of the Workshop on Models of Trust for the Web*, pages 3–131.

Donald Metzler and Tapas Kanungo. 2008. Machine learned sentence selection strategies for query-biased summarization. In *Proceedings of SIGIR Learning to Rank Workshop*.

Besiki Stvilia, Michael B. Twidale, Linda C. Smith, and Les Gasser. 2005. Assessing information quality of a community-based encyclopedia. In *Proceedings of the International Conference on Information Quality*.

Maggy Anastasia Suryanto, Ee Peng Lim, Aixin Sun, and Roger H. L. Chiang. 2009. Quality-aware collaborative question answering: methods and evaluation. In *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 142–151, New York, NY, USA. ACM.

Ashwin Swaminathan, Cherian V. Mathew, and Darko Kirovski. 2009. Essential pages. In *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 173–182, Washington, DC, USA. IEEE Computer Society.

Changhu Wang, Feng Jing, Lei Zhang, and Hong-Jiang Zhang. 2007. Learning query-biased web page summarization. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 555–562, New York, NY, USA. ACM.

Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314, New York, NY, USA. ACM.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009a. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 187–194, New York, NY, USA. ACM.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009b. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 179–186, New York, NY, USA. ACM.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 985–992, Morristown, NJ, USA. Association for Computational Linguistics.

Honglei Zeng, Maher A. Alhossaini, Li Ding, Richard Fikes, and Deborah L. McGuinness. 2006. Computing trust from revision history. In *PST '06: Proceedings of the 2006 International Conference on Privacy, Security and Trust*, pages 1–1, New York, NY, USA. ACM.

Liang Zhou, Chin Y. Lin, and Eduard Hovy. 2006. Summarizing answers for complicated questions. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC), Genoa, Italy*.

A hybrid rule/model-based finite-state framework for normalizing SMS messages

Richard Beaufort[1] **Sophie Roekhaut**[2] **Louise-Amélie Cougnon**[1] **Cédrick Fairon**[1]
(1) CENTAL, Université catholique de Louvain – 1348 Louvain-la-Neuve, Belgium
{richard.beaufort,louise-amelie.cougnon,cedrick.fairon}@uclouvain.be
(2) TCTS Lab, Université de Mons – 7000 Mons, Belgium
sophie.roekhaut@umons.ac.be

Abstract

In recent years, research in natural language processing has increasingly focused on normalizing SMS messages. Different well-defined approaches have been proposed, but the problem remains far from being solved: best systems achieve a 11% Word Error Rate. This paper presents a method that shares similarities with both spell checking and machine translation approaches. The normalization part of the system is entirely based on models trained from a corpus. Evaluated in French by 10-fold-cross validation, the system achieves a 9.3% Word Error Rate and a 0.83 BLEU score.

1 Introduction

Introduced a few years ago, Short Message Service (SMS) offers the possibility of exchanging written messages between mobile phones. SMS has quickly been adopted by users. These messages often greatly deviate from traditional spelling conventions. As shown by specialists (Thurlow and Brown, 2003; Fairon et al., 2006; Bieswanger, 2007), this variability is due to the simultaneous use of numerous coding strategies, like phonetic plays (*2m1* read 'demain', "tomorrow"), phonetic transcriptions (*kom* instead of 'comme', "like"), consonant skeletons (*tjrs* for 'toujours', "always"), misapplied, missing or incorrect separators (*j esper* for 'j'espère', "I hope"; *j'croibi1k*, instead of 'je crois bien que', "I am pretty sure that"), etc. These deviations are due to three main factors: the small number of characters allowed per text message by the service (140 bytes), the constraints of the small phones' keypads and, last but not least, the fact that people mostly communicate between friends and relatives in an informal register.

Whatever their causes, these deviations considerably hamper any standard natural language processing (NLP) system, which stumbles against so many Out-Of-Vocabulary words. For this reason, as noted by Sproat et al. (2001), an SMS normalization must be performed *before* a more conventional NLP process can be applied. As defined by Yvon (2008), "SMS normalization consists in rewriting an SMS text using a more conventional spelling, in order to make it more readable for a human or for a machine."

The SMS normalization we present here was developed in the general framework of an SMS-to-speech synthesis system[1]. This paper, however, only focuses on the normalization process.

Evaluated in French, our method shares similarities with both spell checking and machine translation. The machine translation-like module of the system performs the true normalization task. It is entirely based on models learned from an SMS corpus and its transcription, aligned at the character-level in order to get parallel corpora. Two spell checking-like modules surround the normalization module. The first one detects unambiguous tokens, like URLs or phone numbers, to keep them out of the normalization. The second one, applied on the normalized parts only, identifies non-alphabetic sequences, like punctuations, and labels them with the corresponding token. This greatly helps the system's print module to follow the basic rules of typography.

This paper is organized as follows. Section 2 proposes an overview of the state of the art. Section 3 presents the general architecture of our system, while Section 4 focuses on how we learn and combine our normalization models. Section 5 evaluates the system and compares it to

[1]The *Vocalise* project.
See cental.fltr.ucl.ac.be/team/projects/vocalise/.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 770–779,
Uppsala, Sweden, 11-16 July 2010.

previous works. Section 6 draws conclusions and considers some future possible improvements of the method.

2 Related work

As highlighted by Kobus et al. (2008b), SMS normalization, up to now, has been handled through three well-known NLP metaphors: spell checking, machine translation and automatic speech recognition. In this section, we only present the pros and cons of these approaches. Their results are given in Section 5, focused on our evaluation.

The spell checking metaphor (Guimier de Neef et al., 2007; Choudhury et al., 2007; Cook and Stevenson, 2009) performs the normalization task on a word-per-word basis. On the assumption that most words should be correct for the purpose of communication, its principle is to keep In-Vocabulary words out of the correction process. Guimier de Neef et al. (2007) proposed a rule-based system that uses only a few linguistic resources dedicated to SMS, like specific lexicons of abbreviations. Choudhury et al. (2007) and Cook and Stevenson (2009) preferred to implement the noisy channel metaphor (Shannon, 1948), which assumes a communication process in which a sender emits the intended message W through an imperfect (noisy) communication channel, such that the sequence O observed by the recipient is a noisy version of the original message. On this basis, the idea is to recover the intended message W hidden behind the sequences of observations O, by maximizing:

$$\begin{aligned} W_{max} &= \arg\max P(W|O) \\ &= \arg\max \frac{P(O|W)\,P(W)}{P(O)} \end{aligned} \quad (1)$$

where $P(O)$ is ignored because constant, $P(O|W)$ models the channel's noise, and $P(W)$ models the language of the source. Choudhury et al. (2007) implemented the noisy channel through a Hidden-Markov Model (HMM) able to handle both graphemic variants and phonetic plays as proposed by (Toutanova and Moore, 2002), while Cook and Stevenson (2009) enhanced the model by adapting the channel's noise $P(O|W, wf)$ according to a list of predefined observed word formations $\{wf\}$: stylistic variation, word clipping, phonetic abbreviations, etc. Whatever the system, the main limitation of the spell checking approach is the excessive confidence it places in word boundaries.

The machine translation metaphor, which is historically the first proposed (Bangalore et al., 2002; Aw et al., 2006), considers the process of normalizing SMS as a translation task from a source language (the SMS) to a target language (its standard written form). This standpoint is based on the observation that, on the one side, SMS messages greatly differ from their standard written forms, and that, on the other side, most of the errors cross word boundaries and require a wide context to be handled. On this basis, Aw et al. (2006) proposed a statistical machine translation model working at the phrase-level, by splitting sentences into their k most probable phrases. While this approach achieves really good results, Kobus et al. (2008b) make the assertion that a phrase-based translation can hardly capture the lexical creativity observed in SMS messages. Moreover, the translation framework, which can handle many-to-many correspondences between sources and targets, exceeds the needs of SMS normalization, where the normalization task is almost deterministic.

Based on this analysis, Kobus et al. (2008b) proposed to handle SMS normalization through an automatic speech recognition (ASR) metaphor. The starting point of this approach is the observation that SMS messages present a lot of phonetic plays that sometimes make the SMS word (*sré*, *mwa*) closer to its phonetic representation ([sʁe], [mwa]) than to its standard written form (*serai*, "will be", *moi*, "me"). Typically, an ASR system tries to discover the best word sequence within a lattice of weighted phonetic sequences. Applied to the SMS normalization task, the ASR metaphor consists in first converting the SMS message into a phone lattice, before turning it into a word-based lattice using a phoneme-to-grapheme dictionary. A language model is then applied on the word lattice, and the most probable word sequence is finally chosen by applying a best-path algorithm on the lattice. One of the advantages of the grapheme-to-phoneme conversion is its intrinsic ability to handle word boundaries. However, this step also presents an important drawback, raised by the authors themselves: it prevents next normalization steps from knowing what graphemes were in the initial sequence.

Our approach, which is detailed in Sections 3 and 4, shares similarities with both the spell checking approach and the machine translation principles, trying to combine the advantages of these methods, while leaving aside their drawbacks: like in spell checking systems, we detect unambiguous units of text as soon as possible and try to rely on word boundaries when they seem *reliable enough*; but like in the machine translation task, our method intrinsically handles word boundaries in the normalization process if needed.

3 Overview of the system

3.1 Tools in use

In our system, all lexicons, language models and sets of rules are compiled into finite-state machines (FSMs) and combined with the input text by *composition* ($\circ$). The reader who is not familiar with FSMs and their fundamental theoretical properties, like composition, is urged to consult the state-of-the-art literature (Roche and Schabes, 1997; Mohri and Riley, 1997; Mohri et al., 2000; Mohri et al., 2001).

We used our own finite-state tools: a finite-state machine library and its associated compiler (Beaufort, 2008). In conformance with the format of the library, the compiler builds finite-state machines from weighted rewrite rules, weighted regular expressions and n-gram models.

3.2 Aims

We formulated four constraints before fixing the system's architecture. First, special tokens, like URLs, phones or currencies, should be identified as soon as possible, to keep them out of the normalization process.

Second, word boundaries should be taken into account, as far as they seem *reliable enough*. The idea, here, is to base the decision on a learning able to catch frequent SMS sequences to include in a dedicated In-Vocabulary (IV) lexicon.

Third, any other SMS sequence should be considered as Out-Of-Vocabulary (OOV), on which in-depth rewritings may be applied.

Fourth, the basic rules of typography and typesettings should be applied on the normalized version of the SMS message.

3.3 Architecture

The architecture depicted in Figure 1 directly relies on these considerations. In short, an SMS message first goes through three SMS modules, which normalize its noisy parts. Then, two standard NLP modules produce a morphosyntactic analysis of the normalized text. A last module, finally, takes advantage of this linguistic analysis either to print a text that follows the basic rules of typography, or to synthesize the corresponding speech signal.

Because this paper focuses on the normalization task, the rest of this section only presents the SMS modules and the "smart print" output. The morphosyntactic analysis, made of state-of-the-art algorithms, is described in (Beaufort, 2008), and the text-to-speech synthesis system we use is presented in (Colotte and Beaufort, 2005).

3.3.1 SMS modules

SMS preprocessing. This module relies on a set of manually-tuned rewrite rules. It identifies paragraphs and sentences, but also some

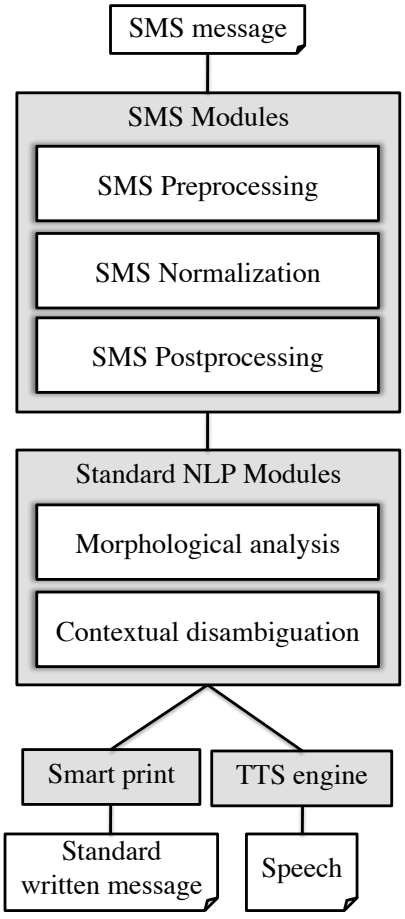


Figure 1: Architecture of the system

unambiguous tokens: URLs, phone numbers, dates, times, currencies, units of measurement and, last but not least in the context of SMS, smileys[2]. These tokens are kept out of the normalization process, while any other sequence of characters is considered – and labelled – as noisy.

SMS normalization. This module only uses models learned from a training corpus (cf. Section 4). It involves three steps. First, an SMS-dedicated lexicon look-up, which differentiates between known and unknown parts of a noisy token. Second, a rewrite process, which creates a lattice of weighted solutions. The rewrite model differs depending on whether the part to rewrite is known or not. Third, a combination of the lattice of solutions with a language model, and the choice of the best sequence of lexical units. At this stage, the normalization as such is completed.

SMS postprocessing. Like the preprocessor, the postprocessor relies on a set of manually-tuned rewrite rules. The module is only applied on the normalized version of the noisy tokens, with the intention to identify any non-alphabetic sequence and to isolate it in a distinct token. At this stage, for instance, a point becomes a 'strong punctuation'. Apart from the list of tokens already managed by the preprocessor, the postprocessor handles as well numeric and alphanumeric strings, fields of data (like bank account numbers), punctuations and symbols.

3.3.2 Smart print

The smart print module, based on manually-tuned rules, checks either the kind of token (chosen by the SMS pre-/post-processing modules) or the grammatical category (chosen by the morphosyntactic analysis) to make the right typography choices, such as the insertion of a space after certain tokens (URLs, phone numbers), the insertion of two spaces after a strong punctuation (point, question mark, exclamation mark), the insertion of two carriage returns at the end of a paragraph, or the upper case of the initial letter at the beginning of the sentence.

[2]Our list contains about 680 smileys.

4 The normalization models

4.1 Overview of the normalization algorithm

Our approach is an approximation of the noisy channel metaphor (cf. Section 2). It differs from this general framework, because we adapt the model of the channel's noise depending on whether the noisy token (our sequence of observations) is In-Vocabulary or Out-Of-Vocabulary:

$$P(O|W) = \begin{cases} P_{IV}(O|W) & \text{if } O \in IV \\ P_{OOV}(O|W) & \text{else} \end{cases} \quad (2)$$

Indeed, our algorithm is based on the assumption that applying different normalization models to IV and OOV words should both improve the results and reduce the processing time.

For this purpose, the first step of the algorithm consists in composing a noisy token T with an FST Sp whose task is to differentiate between sequences of IV words and sequences of OOV words, by labelling them with a special IV or OOV marker. The token is then split in n segments sg_i according to these markers:

$$\{sg\} = \text{Split}(T \circ Sp) \quad (3)$$

In a second step, each segment is composed with a rewrite model according to its kind: the IV rewrite model R_{IV} for sequences of IV words, and the OOV rewrite model R_{OOV} for sequences of OOV words:

$$sg'_i = \begin{cases} sg_i \circ R_{IV} & \text{if } sg_i \in IV \\ sg_i \circ R_{OOV} & \text{else} \end{cases} \quad (4)$$

All rewritten segments are then concatenated together in order to get back the complete token:

$$T = \odot_{i=1}^{n}(sg'_i) \quad (5)$$

where $\odot$ is the concatenation operator.

The third and last normalization step is applied on a complete sentence S. All tokens T_j of S are concatenated together and composed with the lexical language model LM. The result of this composition is a word lattice, of which we take the most probable word sequence S' by applying a best-path algorithm:

$$S' = \text{BestPath}(\ (\odot_{j=1}^{m} T_j) \circ LM\ ) \quad (6)$$

where m is the number of tokens of S. In S', each noisy token T_j of S is mapped onto its most probable normalization.

4.2 The corpus alignment

Our normalization models were trained on a French SMS corpus of 30,000 messages, gathered in Belgium, semi-automatically anonymized and manually normalized by the Catholic University of Louvain (Fairon and Paumier, 2006). Together, the SMS corpus and its transcription constitute *parallel corpora* aligned at the message-level.

However, in order to learn pieces of knowledge from these corpora, we needed a string alignment at the *character-level*.

One way of implementing this string alignment is to compute the edit-distance of two strings, which measures the minimum number of operations (substitutions, insertions, deletions) required to transform one string into the other (Levenshtein, 1966). Using this algorithm, in which each operation gets a cost of 1, two strings may be aligned in different ways with the same global cost. This is the case, for instance, for the SMS form *kozer* ([koze]) and its standard transcription *causé* ("talked"), as illustrated by Figure 2. However, from a linguistic standpoint, alignment (1) is preferable, because corresponding *graphemes* are aligned on their *first character*.

In order to automatically choose this preferred alignment, we had to distinguish the three edit-operations, according to the characters to be aligned. For that purpose, probabilities were required. Computing probabilities for each operation according to the characters to be aligned was performed through an iterative algorithm described in (Cougnon and Beaufort, 2009). In short, this algorithm gradually learns the best way of aligning strings. On our parallel corpora, it converged after 7 iterations and provided us with a result from which the learning could start.

(1)	ko_ser causé_		(2)	k_oser causé_
(3)	ko_ser caus_é		(4)	k_oser caus_é

Figure 2: Different equidistant alignments, using a standard edit-cost of 1. Underscores ('_') mean *insertion* in the upper string, and *deletion* in the lower string.

4.3 The split model Sp

In natural language processing, a word is commonly defined as "a sequence of alphabetic characters between separators", and an IV word is simply a word that belongs to the lexicon in use.

In SMS messages however, separators are surely indicative, but not reliable. For this reason, our definition of the *word* is far from the previous one, and originates from the string alignment. After examining our parallel corpora aligned at the character-level, we decided to consider as a word "the longest sequence of characters parsed without meeting the same separator on both sides of the alignment". For instance, the following alignment

```
J esper_ k___tu va_
J'espère que tu vas
```
(*I hope that you will*)

is split as follows according to our definition:

J esper_ J'espère		k___tu que tu		va_ vas

since the separator in "`J esper`" is different from its transcription, and "`ktu`" does not contain any separator. Thus, this SMS sequence corresponds to 3 SMS words: [*J esper*], [*ktu*] and [*va*].

A first parsing of our parallel corpora provided us with a list of SMS sequences corresponding to our IV lexicon. The FST Sp is built on this basis:

$$Sp = (\ S^*\ (I|O)\ \ (\ S^+(I|O)\ )^*\ S^*\ ) \circ G \quad (7)$$

where:

- I is an FST corresponding to the lexicon, in which IV words are mapped onto the IV marker.
- O is the complement of I[3]. In this OOV lexicon, OOV sequences are mapped onto the OOV marker.
- S is an FST corresponding to the list of separators (any non-alphabetic and non-numeric character), mapped onto a SEP marker.

[3]Actually, the true complement of I accepts sequences with separators, while these sequences were removed from O.

- G is an FST able to detect consecutive sequences of IV (resp. OOV) words, and to group them under a unique IV (resp. OOV) marker. By gathering sequences of IVs and OOVs, SEP markers disappear from Sp.

Figure 3 illustrates the composition of Sp with the SMS sequence *J esper kcv b1* (*J'espère que ça va bien*, "I hope you are well"). For the example, we make the assumption that *kcv* was never seen during the training.

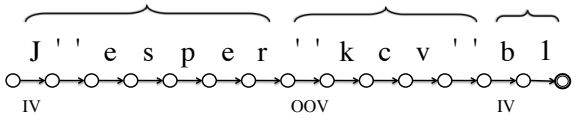


Figure 3: Application of the split model Sp. The OOV sequence starts and ends with separators.

4.4 The IV rewrite model R_{IV}

This model is built during a second parsing of our parallel corpora. In short, the parsing simply gathers all possible normalizations for each SMS sequence put, by the first parsing, in the IV lexicon. Contrary to the first parsing, this second one processes the corpus *without taking separators into account*, in order to make sure that *all* possible normalizations are collected.

Each normalization $\bar{w}$ for a given SMS sequence w is weighted as follows:

$$p(\bar{w}|w) = \frac{\text{Occ}(\bar{w}, w)}{\text{Occ}(w)} \tag{8}$$

where $\text{Occ}(x)$ is the number of occurrences of x in the corpus. The FST R_{IV} is then built as follows:

$$R_{IV} = \ {S_{IV}}^* \ IV_R \ \ (\ {S_{IV}}^+ \ IV_R \ )^* \ {S_{IV}}^* \tag{9}$$

where:

- IV_R is a weighted *lexicon* compiled into an FST, in which each IV sequence is mapped onto the list of its possible normalizations.
- S_{IV} is a weighted *lexicon* of separators, in which each separator is mapped onto the list of its possible normalizations. The deletion is often one of the possible normalization of a separator. Otherwise, the deletion is added and is weighted by the following smoothed probability:

$$p(\texttt{DEL}|w) = \frac{0.1}{\text{Occ}(w) + 0.1} \tag{10}$$

4.5 The OOV rewrite model R_{OOV}

In contrast to the other models, this one is not a regular expression made of weighted lexicons. It corresponds to a set of *weighted rewrite rules* (Chomsky and Halle, 1968; Johnson, 1972; Mohri and Sproat, 1996) learned from the alignment. Developed in the framework of generative phonology, rules take the form

$$\phi \rightarrow \psi : \lambda \ \_ \ \rho \ / \ w \tag{11}$$

which means that the replacement $\phi \rightarrow \psi$ is only performed when ϕ is surrounded by λ on the left and ρ on the right, and gets the weight w. However, in our case, rules take the simpler form

$$\phi \rightarrow \psi \ / \ w \tag{12}$$

which means that the replacement $\phi \rightarrow \psi$ is always performed, whatever the context.

Inputs of our rules (ϕ) are sequences of 1 to 5 characters taken from the SMS side of the alignment, while outputs (ψ) are their corresponding normalizations. Our rules are sorted in the reverse order of the length of their inputs: rules with longer inputs come first in the list.

Long-to-short rule ordering reduces the number of proposed normalizations for a given SMS sequence for two reasons:

1. the firing of a rule with a longer input blocks the firing of any shorter sub-rule. This is due to a constraint expressed on lists of rewrite rules: a given rule may be applied only if no more specific and relevant rule has been met higher in the list;
2. a rule with a longer input usually has fewer alternative normalizations than a rule with a shorter input does, because the longer SMS sequence likely occurred paired with fewer alternative normalizations in the training corpus than did the shorter SMS sequence.

Among the wide set of possible sequences of 2 to 5 characters gathered from the corpus, we only kept in our list of rules the sequences that allowed at least one normalization *solely made of IV words*. It is important to notice that here, we refer to the standard notion of *IV word*: while gathering the candidate sequences from the corpus, we systematically checked each word of the normalizations against a lexicon of French

standard written forms. The lexicon we used contains about 430,000 inflected forms and is derived from Morlex[4], a French lexical database.

Figure 4 illustrates these principles by focusing on 3 input sequences: *aussi*, *au* and *a*. As shown by the Figure, all rules of a set dedicated to the same input sequence (for instance, *aussi*) are optional (?→), except the last one, which is obligatory (→). In our finite-state compiler, this convention allows the application of all concurrent normalizations on the same input sequence, as depicted in Figure 5.

In our real list of OOV rules, the input sequence *a* corresponds to 231 normalizations, while *au* accepts 43 normalizations and *aussi*, only 3. This highlights the interest, in terms of efficiency, of the long-to-short rule ordering.

4.6 The language model

Our language model is an n-gram of lexical forms, smoothed by linear interpolation (Chen and Goodman, 1998), estimated on the normalized part of our training corpus and compiled into a weighted FST LM_w.

At this point, this FST cannot be combined with our other models, because it works on *lexical units* and not on *characters*. This problem is solved by composing LM_w with another FST L, which represents a lexicon mapping each input word, considered as a string of characters, onto the same output words, but considered here as a lexical unit. Lexical units are then permanently removed from the language model by keeping only the first projection (the input side) of the composition:

$$LM = \text{FirstProjection}(\ L \circ LM_w\ ) \qquad (13)$$

In this model, special characters, like punctuations or symbols, are represented by their categories (light, medium and strong punctuations, question mark, symbol, etc.), while special tokens, like URLs or phone numbers, are handled as token values (URL, phone, etc.) instead of as sequences of characters. This reduces the complexity of the model.

As we explained earlier, tokens of a same sentence S are concatenated together at the end of the second normalization step. During this concatenation process, sequences corresponding to special tokens are automatically replaced by their token values. Special characters, however,

[4]See http://bach.arts.kuleuven.be/pmertens/.

```
"aussi" ?-> "au si" / 8.4113 (*)
"aussi" ?-> "ou si" / 6.6743 (*)
"aussi" ->  "aussi" / 0.0189 (*)
...
...
"au" ?-> "ow" / 14.1787
...
"au" ?-> "ôt" / 12.5938
"au" ?-> "du" / 12.1787 (*)
"au" ?-> "o" / 11.8568
...
"au" ?-> "on" / 10.8568 (*)
...
"au" ?-> "aud" / 9.9308
"au" ?-> "aux" / 6.1731 (*)
"au" ->  "au" / 0.0611 (*)
...
...
"a" ?-> "a d" / 17.8624
"a" ?-> "ation" / 17.8624
"a" ?-> "âts" / 17.8624
...
"a" ?-> "ablement" / 16.8624
"a" ?-> "anisation" / 16.8624
...
"a" ?-> "u" / 15.5404
"a" ?-> "y a" / 15.5404
...
"a" ?-> "abilité" / 13.4029
"a" ?-> "à-" / 12.1899
"a" ?-> "ar" / 11.5225
"a" ?-> \DEL / 9.1175
"a" ?-> "ça" / 6.2019
"a" ?-> "à" / 3.5013
"a" ->  "a" / 0.3012
```

Figure 4: Samples from the list of OOV rules. Rules' weights are negative logarithms of probabilities: smaller weights are thus better. Asterisks indicate normalizations solely made of French IV words.

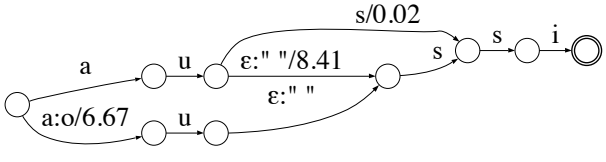


Figure 5: Application of the OOV rules on the input sequence *aussi*. All normalizations corresponding to this sequence were allowed, while rules corresponding to shorter input sequences were ignored.

are still present in S. For this reason, S is first composed with an FST $Reduce$, which maps each special character onto its corresponding category:

$$S \circ Reduce \circ LM \quad (14)$$

5 Evaluation

The performance and the efficiency of our system were evaluated on a MacBook Pro with a 2.4 GHz Intel Core 2 Duo CPU, 4 GB 667 MHz DDR2 SDRAM, running Mac OS X version 10.5.8.

The evaluation was performed on the corpus of 30,000 French SMS presented in Section 4.2, by ten-fold cross-validation (Kohavi, 1995). The principle of this method of evaluation is to split the initial corpus into 10 subsets of equal size. The system is then trained 10 times, each time leaving out one of the subsets from the training corpus, but using only this omitted subset as test corpus.

The language model of the evaluation is a 3-gram. We did not try a 4-gram. This choice was motivated by the experiments of Kobus et al. (2008a), who showed on a French corpus comparable to ours that, if using a larger language model is always rewarded, the improvement quickly decreases with every higher level and is already quite small between 2-gram and 3-gram.

Table 1 presents the results in terms of efficiency. The system seems efficient, while we cannot compare it with other methods, which did not provide us with this information.

Table 2, part 1, presents the performance of our approach (*Hybrid*) and compares it to a trivial copy-paste (*Copy*). The system was evaluated in terms of BLEU score (Papineni et al., 2001), Word Error Rate (WER) and Sentence Error Rate (SER). Concerning WER, the table presents the distribution between substitutions (Sub), deletions (Del) and insertions (Ins). The copy-paste results just inform about the real deviation of our corpus from the traditional spelling conventions, and highlight the fact that our system is still at pains to significantly reduce the SER, while results in terms of WER and BLEU score are quite encouraging.

Table 2, part 2, provides the results of the state-of-the-art approaches. The only results truly comparable to ours are those of Guimier de Neef et al. (2007), who evaluated their approach on the same corpus as ours[5]; clearly, our method outperforms theirs. Our results also seem a bit better than those of Kobus et al. (2008a), although the comparison with this system, also evaluated in French, is less easy: they combined the French corpus we used with another one and performed a single validation, using a bigger training corpus (36.704 messages) for a test corpus quite similar to one of our subsets (2.998 SMS). Other systems were evaluated in English, and results are more difficult to compare; at least, our results seem in line with them.

	mean	**dev.**
bps	1836.57	159.63
ms/SMS (140b)	76.23	22.34

Table 1: Efficiency of the system.

The analysis of the normalizations produced by our system pointed out that, most often, errors are contextual and concern the gender (*quel(le)*, "what"), the number (*bisou(s)*, "kiss"), the person (*[tu t']inquiète(s)*, "you are worried") or the tense (*arrivé/arriver*, "arrived"/"to arrive"). That contextual errors are frequent is not surprising. In French, as mentioned by Kobus et al. (2008b), n-gram models are unable to catch this information, as it is generally out of their scope.

On the other hand, this analysis confirmed our initial assumptions. First, special tokens (URLs, phones, etc.) are not modified. Second, agglutinated words are generally split (*Pensa ms* → *Pense à mes*, "think to my"), while misapplied separators tend to be deleted (*G t* → *J'étais*, "I was"). Of course, we also found some errors at word boundaries (*[il] l'arrange* → *[il] la range*, "[he] arranges" → "[he] pits in order"), but they were fairly rare.

6 Conclusion and perspectives

In this paper, we presented an SMS normalization framework based on finite-state machines and developed in the context of an SMS-to-speech synthesis system. With the intention to avoid wrong modifications of special tokens and to handle word boundaries as easily as possible, we designed a method that shares similarities with both spell checking and machine translation. Our

[5]They performed an evaluation without ten-fold cross-validation, because their rule-based system did not need any training.

	1. Our approach				2. State of the art					
	Ten-fold cross-validation, French				French			English		
	Copy		*Hybrid*		*Guimier*	*Kobus 2008*		*Aw*	*Choud.*	Cook
	$\bar{x}$	σ	$\bar{x}$	σ	*2007*	1	2*	*2006*	*2006***	*2009***
Sub.	25.90	1.65	6.69	0.45		11.94				
Del.	8.24	0.74	1.89	0.31		2.36				
Ins.	0.46	0.08	0.72	0.10		2.21				
WER	34.59	2.37	**9.31**	0.78		16.51	**10.82**		41.00	44.60
SER	85.74	0.87	**65.07**	1.85		76.05				
BLEU	0.47	0.03	**0.83**	0.01	**0.736**		**0.8**	**0.81**		

$\bar{x}$=*mean,* σ=*standard deviation*

Table 2: Performance of the system. (*) Kobus 2008-1 corresponds to the ASR-like system, while Kobus 2008-2 is a combination of this system with a series of open-source machine translation toolkits. (**) Scores obtained on noisy data only, out of the sentence's context.

normalization algorithm is original in two ways. First, it is entirely based on models learned from a training corpus. Second, the rewrite model applied to a noisy sequence differs depending on whether this sequence is known or not.

Evaluated by ten-fold cross-validation, the system seems efficient, and the performance in terms of BLEU score and WER are quite encouraging. However, the SER remains too high, which emphasizes the fact that the system needs several improvements.

First of all, the model should take phonetic similarities into account, because SMS messages contain a lot of phonetic plays. The phonetic model, for instance, should know that *o*, *au*, *eau*, ..., *aux* can all be pronounced [o], while *è*, *ais*, *ait*, ..., *aient* are often pronounced [ɛ]. However, unlike Kobus et al. (2008a), we feel that this model must avoid the normalization step in which the graphemic sequence is converted into phonemes, because this conversion prevents the next steps from knowing which graphemes were in the initial sequence. Instead, we propose to *learn* phonetic similarities from a dictionary of words with phonemic transcriptions, and to build *graphemes-to-graphemes* rules. These rules could then be automatically weighted, by learning their frequencies from our aligned corpora. Furthermore, this model should be able to allow for timbre variation, like [e]–[ɛ], in order to allow similarities between graphemes frequently confused in French, like *ai* ([e]) and *ais*/*ait*/*aient* ([ɛ]). Last but not least, the graphemes-to-graphemes rules should be contextualized, in order to reduce the complexity of the model.

It would also be interesting to test the impact of another lexical language model, learned on non-SMS sentences. Indeed, the lexical model must be learned from sequences of *standard written forms*, an obvious prerequisite that involves a major drawback when the corpus is made of SMS sentences: the corpus must first be *transcribed*, an expensive process that reduces the amount of data on which the model will be trained. For this reason, we propose to learn a lexical model from non-SMS sentences. However, the corpus of external sentences should still share two important features with the SMS language: it should mimic the oral language and be as spontaneous as possible. With this in mind, our intention is to gather sentences from Internet forums. But not just any forum, because often forums share another feature with the SMS language: their language is noisy. Thus, the idea is to choose a forum asking its members to pay attention to spelling mistakes and grammatical errors, and to avoid the use of the SMS language.

Acknowledgments

This research was funded by grants no. 716619 and 616422 from the Walloon Region of Belgium, and supported by the Multitel research centre.

We sincerely thank our anonymous reviewers for their insightful and helpful comments on the first version of this paper.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text

normalization. In *Proc. COLING/ACL 2006*.

Srinivas Bangalore, Vanessa Murdock, and Giuseppe Riccardi. 2002. Bootstrapping bilingual data using consensus translation for a multilingual instant messaging system. In *Proc. the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA.

Richard Beaufort. 2008. *Application des machines à etats finis en synthèse de la parole. Sélection d'unités non uniformes et correction orthographique*. Ph.D. thesis, FUNDP, Namur, Belgium, March. 605 pages.

Markus Bieswanger. 2007. abbrevi8 or not 2 abbrevi8: A contrastive analysis of different space and time-saving strategies in English and German text messages. In *Texas Linguistics Forum*, volume 50.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report 10-98, Computer Science Group, Harvard University.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English*. Harper and Row, New York, NY.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar1, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Vincent Colotte and Richard Beaufort. 2005. Linguistic features weighting for a text-to-speech system without prosody model. In *Proc. Interspeech'05*, pages 2549–2552.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. In *Proc. Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78.

Louise-Amélie Cougnon and Richard Beaufort. 2009. SSLD: a French SMS to standard language dictionary. In Sylviane Granger and Magali Paquot, editors, *Proc. eLexicography in the 21st century: New applications, new challenges (eLEX 2009)*. Presses Universitaires de Louvain. To appear.

Cédrick Fairon and Sébastien Paumier. 2006. A translated corpus of 30,000 French SMS. In *Proc. LREC 2006*, May.

Cécrick. Fairon, Jean R. Klein, and Sébastien Paumier. 2006. *Le langage SMS: étude d'un corpus informatisé à partir de l'enquête Faites don de vos SMS à la science*. Presses Universitaires de Louvain. 136 pages.

Emilie Guimier de Neef, Arnaud Debeurme, and Jungyeul Park. 2007. TILT correcteur de SMS: évaluation et bilan quantitatif. In *Actes de TALN 2007*, pages 123–132, Toulouse, France.

C. Douglas Johnson. 1972. *Formal aspects of phonological description*. Mouton, The Hague.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008a. Normalizing SMS: are two metaphors better than one? In *Proc. COLING 2008*, pages 441–448, Manchester, UK.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008b. Transcrire les SMS comme on reconnaît la parole. In *Actes de la Conférence sur le Traitement Automatique des Langues (TALN'08)*, pages 128–138, Avignon, France.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI'95*, pages 1137–1143.

Vladimir Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics*, 10:707–710.

Mehryar Mohri and Michael Riley. 1997. Weighted determinization and minimization for large vocabulary speech recognition. In *Proc. Eurospeech'97*, pages 131–134.

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Proc. ACL'96*, pages 231–238.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2001. Generic ϵ-removal algorithm for weighted automata. *Lecture Notes in Computer Science*, 2088:230–242.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL 2001*, pages 311–318.

Emmanuel Roche and Yves Schabes, editors. 1997. *Finite-state language processing*. MIT Press, Cambridge.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

Richard Sproat, A.W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Crispin Thurlow and Alex Brown. 2003. Generation txt? The sociolinguistics of young people's text-messaging. *Discourse Analysis Online*, 1(1).

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proc. ACL'02*, pages 144–151.

François Yvon. 2008. Reorthography of SMS messages. Technical Report 2008, LIMSI/CNRS, Orsay, France.

Letter-Phoneme Alignment: An Exploration

Sittichai Jiampojamarn and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{sj,kondrak}@cs.ualberta.ca

Abstract

Letter-phoneme alignment is usually generated by a straightforward application of the EM algorithm. We explore several alternative alignment methods that employ phonetics, integer programming, and sets of constraints, and propose a novel approach of refining the EM alignment by aggregation of best alignments. We perform both intrinsic and extrinsic evaluation of the assortment of methods. We show that our proposed EM-Aggregation algorithm leads to the improvement of the state of the art in letter-to-phoneme conversion on several different data sets.

1 Introduction

Letter-to-phoneme (L2P) conversion (also called grapheme-to-phoneme conversion) is the task of predicting the pronunciation of a word given its orthographic form by converting a sequence of letters into a sequence of phonemes. The L2P task plays a crucial role in speech synthesis systems (Schroeter et al., 2002), and is an important part of other applications, including spelling correction (Toutanova and Moore, 2001) and speech-to-speech machine translation (Engelbrecht and Schultz, 2005). Many data-driven techniques have been proposed for letter-to-phoneme conversion systems, including neural networks (Sejnowski and Rosenberg, 1987), decision trees (Black et al., 1998), pronunciation by analogy (Marchand and Damper, 2000), Hidden Markov Models (Taylor, 2005), and constraint satisfaction (Bosch and Canisius, 2006).

Letter-phoneme alignment is an important step in the L2P task. The training data usually consists of pairs of letter and phoneme sequences, which are not aligned. Since there is no explicit information indicating the relationships between individual letter and phonemes, these must be inferred by a letter-phoneme alignment algorithm before a prediction model can be trained. The quality of the alignment affects the accuracy of L2P conversion. Letter-phoneme alignment is closely related to transliteration alignment (Pervouchine et al., 2009), which involves graphemes representing different writing scripts. Letter-phoneme alignment may also be considered as a task in itself; for example, in the alignment of speech transcription with text in spoken corpora.

Most previous L2P approaches induce the alignment between letters and phonemes with the expectation maximization (EM) algorithm. In this paper, we propose a number of alternative alignment methods, and compare them to the EM-based algorithms using both intrinsic and extrinsic evaluations. The intrinsic evaluation is conducted by comparing the generated alignments to a manually-constructed gold standard. The extrinsic evaluation uses two different generation techniques to perform letter-to-phoneme conversion on several different data sets. We discuss the advantages and disadvantages of various methods, and show that better alignments tend to improve the accuracy of the L2P systems regardless of the actual technique. In particular, one of our proposed methods advances the state of the art in L2P conversion. We also examine the relationship between alignment entropy and alignment quality.

This paper is organized as follows. In Section 2, we enumerate the assumptions that the alignment methods commonly adopt. In Section 3, we review previous work that employs the EM approach. In Sections 4, 5 and 6, we describe alternative approaches based on phonetics, manually-constructed constraints, and Integer Programming, respectively. In Section 7, we propose an algorithm to refine the alignments produced by EM. Sections 8 and 9 are devoted to the intrinsic and extrinsic evaluation of various approaches. Section 10 concludes the paper.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 780–788,
Uppsala, Sweden, 11-16 July 2010.

2 Background

We define the letter-phoneme alignment task as the problem of inducing *links* between units that are related by pronunciation. Each link is an instance of a specific *mapping* between letters and phonemes. The leftmost example alignment of the word *accuse* [@kjuz] below includes 1-1, 1-0, 1-2, and 2-1 links. The letter e is considered to be linked to special *null phoneme*.

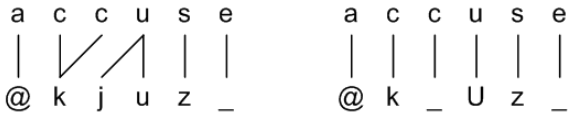


Figure 1: Two alignments of *accuse*.

The following constraints on links are assumed by some or all alignment models:

- the *monotonicity* constraint prevents links from crossing each other;
- the *representation* constraint requires each phoneme to be linked to at least one letter, thus precluding nulls on the letter side;
- the *one-to-one* constraint stipulates that each letter and phoneme may participate in at most one link.

These constraints increasingly reduce the search space and facilitate the training process for the L2P generation models.

We refer to an alignment model that assumes all three constraints as a pure one-to-one (1-1) model. By allowing only 1-1 and 1-0 links, the alignment task is thus greatly simplified. In the simplest case, when the number of letters is equal to the number of phonemes, there is only one possible alignment that satisfies all three constraints. When there are more letters than phonemes, the search is reduced to identifying letters that must be linked to null phonemes (the process referred to as "epsilon scattering" by Black et al. (1998)). In some words, however, one letter clearly represents more than one phoneme; for example, u in Figure 1. Moreover, a pure 1-1 approach cannot handle cases where the number of phonemes exceeds the number of letters. A typical solution to overcome this problems is to introduce so-called *double phonemes* by merging adjacent phonemes that could be represented as a single letter. For example, a double phoneme U would replace a sequence of the phonemes j and u in Figure 1. This solution requires a manual extension of the set of phonemes present in the data. By convention, we regard the models that include a restricted set of 1-2 mappings as 1-1 models.

Advanced L2P approaches, including the joint n-gram models (Bisani and Ney, 2008) and the joint discriminative approach (Jiampojamarn et al., 2007) eliminate the one-to-one constraint entirely, allowing for linking of multiple letters to multiple phonemes. We refer to such models as many-to-many (M-M) models.

3 EM Alignment

Early EM-based alignment methods (Daelemans and Bosch, 1997; Black et al., 1998; Damper et al., 2005) were generally pure 1-1 models. The 1-1 alignment problem can be formulated as a dynamic programming problem to find the maximum score of alignment, given a probability table of aligning letter and phoneme as a mapping function. The dynamic programming recursion to find the most likely alignment is the following:

$$C_{i,j} = \max \begin{cases} C_{i-1,j-1} + \delta(x_i, y_j) \\ C_{i-1,j} + \delta(x_i, \epsilon) \\ C_{i,j-1} + \delta(\epsilon, y_j) \end{cases} \quad (1)$$

where $\delta(x_i, \epsilon)$ denotes a probability that a letter x_i aligns with a null phoneme and $\delta(\epsilon, y_j)$ denotes a probability that a null letter aligns with a phoneme y_j. In practice, the latter probability is often set to zero in order to enforce the representation constraint, which facilitates the subsequent phoneme generation process. The probability table $\delta(x_i, y_j)$ can be initialized by a uniform distribution and is iteratively re-computed (M-step) from the most likely alignments found at each iteration over the data set (E-step). The final alignments are constructed after the probability table converges.

M2M-aligner (Jiampojamarn et al., 2007) is a many-to-many (M-M) alignment algorithm based on EM that allows for mapping of multiple letters to multiple phonemes. Algorithm 1 describes the E-step of the many-to-many alignment algorithm. γ represents partial counts collected over all possible mappings between substrings of letters and phonemes. The maximum lengths of letter and phoneme substrings are controlled by the

Algorithm 1: Many-to-many alignment

Input: $x, y, maxX, maxY, \gamma$
Output: γ

1 $\alpha :=$ FORWARD-M2M $(x, y, maxX, maxY)$
2 $\beta :=$ BACKWARD-M2M $(x, y, maxX, maxY)$
3 $T = |x| + 1$, $V = |y| + 1$
4 **if** $(\alpha_{T,V} = 0)$ **then**
5 return
6 **for** $t = 1..T$, $v = 1..V$ **do**
7 **for** $i = 1..maxX$ **st** $t - i \geq 0$ **do**
8 $\gamma(x_{t-i+1}^{t}, \epsilon) \mathrel{+}= \frac{\alpha_{t-i,v}\delta(x_{t-i+1}^{t}, \epsilon)\beta_{t,v}}{\alpha_{T,V}}$
9 **for** $i = 1..maxX$ **st** $t - i \geq 0$ **do**
10 **for** $j = 1..maxY$ **st** $v - j \geq 0$ **do**
11 $\gamma(x_{t-i+1}^{t}, y_{v-j+1}^{v}) \mathrel{+}= \frac{\alpha_{t-i,v-j}\delta(x_{t-i+1}^{t}, y_{v-j+1}^{v})\beta_{t,v}}{\alpha_{T,V}}$

$maxX$ and $maxY$ parameters. The forward probability α is estimated by summing the probabilities from left to right, while the backward probability β is estimated in the opposite direction. The FORWARD-M2M procedure is similar to line 3 to 10 of Algorithm 1, except that it uses Equation 2 in line 8 and 3 in line 11. The BACKWARD-M2M procedure is analogous to FORWARD-M2M.

$$\alpha_{t,v} \mathrel{+}= \delta(x_{t-i+1}^{t}, \epsilon)\alpha_{t-i,v} \quad (2)$$

$$\alpha_{t,v} \mathrel{+}= \delta(x_{t-i+1}^{t}, y_{v-j+1}^{v})\alpha_{t-i,v-j} \quad (3)$$

In M-step, the partial counts are normalized by using a conditional distribution to create the mapping probability table δ. The final many-to-many alignments are created by finding the most likely paths using the Viterbi algorithm based on the learned mapping probability table. The source code of M2M-aligner is publicly available.[1]

Although the many-to-many approach tends to create relatively large models, it generates more intuitive alignments and leads to improvement in the L2P accuracy (Jiampojamarn et al., 2007). However, since many links involve multiple letters, it also introduces additional complexity in the phoneme prediction phase. One possible solution is to apply a letter segmentation algorithm at test time to cluster letters according to the alignments in the training data. This is problematic because of error propagation inherent in such a process. A better solution is to combine segmentation and decoding using a phrasal decoder (e.g. (Zens and Ney, 2004)).

[1] `http://code.google.com/p/m2m-aligner/`

4 Phonetic alignment

The EM-based approaches to L2P alignment treat both letters and phonemes as abstract symbols. A completely different approach to L2P alignment is based on the phonetic similarity between phonemes. The key idea of the approach is to represent each letter by a phoneme that is likely to be represented by the letter. The actual phonemes on the phoneme side and the phonemes representing letters on the letter side can then be aligned on the basis of phonetic similarity between phonemes. The main advantage of the phonetic alignment is that it requires no training data, and so can be readily be applied to languages for which no pronunciation lexicons are available.

The task of identifying the phoneme that is most likely to be represented by a given letter may seem complex and highly language-dependent. For example, the letter a can represent no less than 12 different English vowels. In practice, however, absolute precision is not necessary. Intuitively, the letters that had been chosen (often centuries ago) to represent phonemes in any orthographic system tend to be close to the prototype phoneme in the original script. For example, the letter 'o' represented a mid-high rounded vowel in Classical Latin and is still generally used to represent similar vowels.

The following simple heuristic works well for a number of languages: treat every letter as if it were a symbol in the International Phonetic Alphabet (IPA). The set of symbols employed by the IPA includes the 26 letters of the Latin alphabet, which tend to correspond to the phonemes that they represent in the Latin script. For example, the IPA symbol [m] denotes a voiced bilabial nasal consonant, which is the phoneme represented by the letter m in most languages that utilize Latin script.

ALINE (Kondrak, 2000) performs phonetic alignment of two strings of phonemes. It combines a dynamic programming alignment algorithm with an appropriate scoring scheme for computing phonetic similarity on the basis of multivalued features. The example below shows the alignment of the word *sheath* to its phonetic transcription [ʃiθ]. ALINE correctly links the most similar pairs of phonemes (s:ʃ, e:i, t:θ).[2]

[2] ALINE can also be applied to non-Latin scripts by replacing every grapheme with the IPA symbol that is phonetically closest to it.

s	h	e	a	t	h
|	|	|	|	|	|
ʃ	-	i	-	θ	-

Since ALINE is designed to align phonemes with phonemes, it does not incorporate the representation constraint. In order to avoid the problem of unaligned phonemes, we apply a post-processing algorithm, which also handles 1-2 links. The algorithm first attempts to remove 0-1 links by merging them with the adjacent 1-0 links. If this is not possible, the algorithm scans a list of valid 1-2 mappings, attempting to replace a pair of 0-1 and 1-1 links with a single 1-2 link. If this also fails, the entire entry is removed from the training set. Such entries often represent unusual foreign-origin words or outright annotation errors. The number of unaligned entries rarely exceeds 1% of the data.

The post-processing algorithm produces an alignment that contains 1-0, 1-1, and 1-2 links. The list of valid 1-2 mappings must be prepared manually. The length of such lists ranges from 1 for Spanish and German (x:[ks]) to 17 for English. This approach is more robust than the double-phoneme technique because the two phonemes are clustered only if they can be linked to the corresponding letter.

5 Constraint-based alignment

One of the advantages of the phonetic alignment is its ability to rule out phonetically implausible letter-phoneme links, such as o:p. We are interested in establishing whether a set of allowable letter-phoneme mappings could be derived directly from the data without relying on phonetic features.

Black et al. (1998) report that constructing lists of possible phonemes for each letter leads to L2P improvement. They produce the lists in a "semi-automatic", interactive manner. The lists constrain the alignments performed by the EM algorithm and lead to better-quality alignments.

We implement a similar interactive program that incrementally expands the lists of possible phonemes for each letter by refining alignments constrained by those lists. However, instead of employing the EM algorithm, we induce alignments using the standard edit distance algorithm with substitution and deletion assigned the same cost. In cases when there are multiple alternative alignments that have the same edit distance, we randomly choose one of them. Furthermore, we extend this idea also to many-to-many alignments. In addition to lists of phonemes for each letter (1-1 mappings), we also construct lists of many-to-many mappings, such as ee:i, sch:ʃ, and ew:ju. In total, the English set contains 377 mappings, of which more than half are of the 2-1 type.

6 IP Alignment

The process of manually inducing allowable letter-phoneme mappings is time-consuming and involves a great deal of language-specific knowledge. The Integer Programming (IP) framework offers a way to induce similar mappings without a human expert in the loop. The IP formulation aims at identifying the smallest set of letter-phoneme mappings that is sufficient to align all instances in the data set.

Our IP formulation employs the three constraints enumerated in Section 2, except that the one-to-one constraint is relaxed in order to identify a small set of 1-2 mappings. We specify two types of binary variables that correspond to local alignment links and global letter-phoneme mappings, respectively. We distinguish three types of local variables, X, Y, and Z, which correspond to 1-0, 1-1, and 1-2 links, respectively. In order to minimize the number of global mappings, we set the following objective that includes variables corresponding to 1-1 and 1-2 mappings:

$$\text{minimize} : \sum_{l,p} G(l,p) + \sum_{l,p_1,p_2} G(l,p_1p_2) \quad (4)$$

We adopt a simplifying assumption that any letter can be linked to a null phoneme, so no global variables corresponding to 1-0 mappings are necessary.

In the lexicon entry k, let l_{ik} be the letter at position i, and p_{jk} the phoneme at position j. In order to prevent the alignments from utilizing letter-phoneme mappings which are not on the global list, we impose the following constraints:

$$\forall_{i,j,k} Y(i,j,k) \leq G(l_{ik}, p_{jk}) \quad (5)$$

$$\forall_{i,j,k} Z(i,j,k) \leq G(l_{ik}, p_{jk}p_{(j+1)k}) \quad (6)$$

For example, the local variable $Y(i,j,k)$ is set if l_{ik} is linked to p_{jk}. A corresponding global variable $G(l_{ik}, p_{jk})$ is set if the list of allowed letter-phoneme mappings includes the link (l_{ik}, p_{jk}). Activating the local variable implies activating the corresponding global variable, but not vice versa.

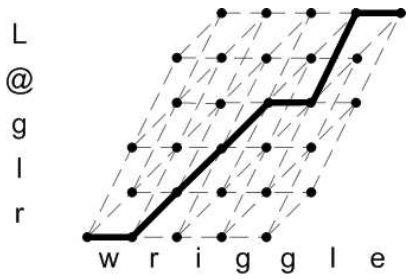


Figure 2: A network of possible alignment links.

We create a network of possible alignment links for each lexicon entry k, and assign a binary variable to each link in the network. Figure 2 shows an alignment network for the lexicon entry k: *wriggle* [r I g @ L]. There are three 1-0 links (level), three 1-1 links (diagonal), and one 1-2 link (steep). The local variables that receive the value of 1 are the following: X(1,0,k), Y(2,1,k), Y(3,2,k), Y(4,3,k), X(5,3,k), Z(6,5,k), and X(7,5,k). The corresponding global variables are: G(r,r), G(i,I), G(g,g), and G(l,@L).

We create constraints to ensure that the link variables receiving a value of 1 form a left-to-right path through the alignment network, and that all other link variables receive a value of 0. We accomplish this by requiring the sum of the links entering each node to equal the sum of the links leaving each node.

$$\forall_{i,j,k} \quad X(i,j,k) + Y(i,j,k) + Z(i,j,k) = X(i+1,j,k) + Y(i+1,j+1,k) + Z(i+1,j+2,k)$$

We found that inducing the IP model with the full set of variables gives too much freedom to the IP program and leads to inferior results. Instead, we first run the full set of variables on a subset of the training data which includes only the lexicon entries in which the number of phonemes exceeds the number of letters. This generates a small set of plausible 1-2 mappings. In the second pass, we run the model on the full data set, but we allow only the 1-2 links that belong to the initial set of 1-2 mappings induced in the first pass.

6.1 Combining IP with EM

The set of allowable letter-phoneme mappings can also be used as an input to the EM alignment algorithm. We call this approach *IP-EM*. After inducing the minimal set of letter-phoneme mappings, we constrain EM to use only those mappings with the exclusion of all others. We initialize the probability of the minimal set with a uniform distribution, and set it to zero for other mappings. We train the EM model in a similar fashion to the many-to-many alignment algorithm presented in Section 3, except that we limit the letter size to be one letter, and that any letter-phoneme mapping that is not in the minimal set is assigned zero count during the E-step. The final alignments are generated after the parameters converge.

7 Alignment by aggregation

During our development experiments, we observed that the technique that combines IP with EM described in the previous section generally leads to alignment quality improvement in comparison with the IP alignment. Nevertheless, because EM is constrained not to introduce any new letter-phoneme mappings, many incorrect alignments are still proposed. We hypothesized that instead of pre-constraining EM, a post-processing of EM's output may lead to better results.

M2M-aligner has the ability to create precise links involving more than one letter, such as *ph*:f. However, it also tends to create non-intuitive links such as *se*:z for the word *phrase* [f r e z], where e is clearly a case of a "silent" letter. We propose an alternative EM-based alignment method that instead utilizes a list of alternative *one-to-many* alignments created with M2M-aligner and aggregates 1-M links into M-M links in cases when there is a disagreement between alignments within the list. For example, if the list contains the two alignments shown in Figure 3, the algorithm creates a single many-to-many alignment by merging the first pair of 1-1 and 1-0 links into a single *ph:f* link. However, the two rightmost links are *not* merged because there is no disagreement between the two initial alignments. Therefore, the resulting alignment reinforces the *ph*:f mapping, but avoids the questionable *se*:z link.

```
p h r a s e      p h r a s e
| | | | | |      | | | | | |
f - r e z -      - f r e z -
```

Figure 3: Two alignments of *phrase*.

In order to generate the list of best alignments, we use Algorithm 2, which is an adaptation of the standard Viterbi algorithm. Each cell $Q_{t,v}$ contains a list of n-best scores that correspond to al-

Algorithm 2: Extracting n-best alignments

Input: x, y, δ
Output: $Q_{T,V}$

```
1  T = |x| + 1 , V = |y| + 1
2  for t = 1..T do
3      Q_{t,v} = ∅
4      for v = 1..V do
5          for q ∈ Q_{t-1,v} do
6              append q · δ(x_t, ϵ) to Q_{t,v}
7          for j = 1..maxY st v − j ≥ 0 do
8              for q ∈ Q_{t-1,v-j} do
9                  append q · δ(x_t, y^v_{v-j+1}) to Q_{t,v}
10         sort Q_{t,v}
11         Q_{t,v} = Q_{t,v}[1 : n]
```

ternative alignments during the forward pass. In line 9, we consider all possible 1-M links between letter x_t and phoneme substring y^v_{v-j+1}. At the end of the main loop, we keep at most n best alignments in each $Q_{t,v}$ list.

Algorithm 2 yields n-best alignments in the $Q_{T,V}$ list. However, in order to further restrict the set of high-quality alignments, we also discard the alignments with scores below threshold R with respect to the best alignment score. Based on the experiments with the development set, we set $R = 0.8$ and $n = 10$.

8 Intrinsic evaluation

For the intrinsic evaluation, we compared the generated alignments to gold standard alignments extracted from the the core vocabulary of the Combilex data set (Richmond et al., 2009). Combilex is a high quality pronunciation lexicon with explicit expert manual alignments. We used a subset of the lexicon composed of the core vocabulary containing 18,145 word-phoneme pairs. The alignments contain 550 mappings, which include complex 4-1 and 2-3 types.

Each alignment approach creates alignments from unaligned word-phoneme pairs in an unsupervised fashion. We distinguish between the 1-1 and M-M approaches. We report the alignment quality in terms of precision, recall and F-score. Since the gold standard includes many links that involve multiple letters, the theoretical upper bound for recall achieved by a one-to-one approach is 90.02%. However, it is possible to obtain the perfect precision because we count as correct all 1-1 links that are *consistent* with the M-M links in the gold standard. The F-score corresponding to perfect precision and the upper-bound recall is 94.75%.

Alignment entropy is a measure of alignment quality proposed by Pervouchine et al. (2009) in the context of transliteration. The entropy indicates the uncertainty of mapping between letter l and phoneme p resulting from the alignment: We compute the alignment entropy for each of the methods using the following formula:

$$H = -\sum_{l,p} P(l,p) \log P(l|p) \qquad (7)$$

Table 1 includes the results of the intrinsic evaluation. (the two rightmost columns are discussed in Section 9). The baseline *BaseEM* is an implementation of the one-to-one alignment method of (Black et al., 1998) *without* the allowable list. *ALINE* is the phonetic method described in Section 4. *SeedMap* is the hand-seeded method described in Section 5. *M-M-EM* is the M2M-aligner approach of Jiampojamarn et al. (2007). *1-M-EM* is equivalent to *M-M-EM* but with the restriction that each link contains exactly one letter. *IP-align* is the alignment generated by the IP formulation from Section 6. *IP-EM* is the method that combines IP with EM described in Section 6.1. *EM-Aggr* is our final many-to-many alignment method described in Section 7. *Oracle* corresponds to the gold-standard alignments from Combilex.

Overall, the M-M models obtain lower precision but higher recall and F-score than 1-1 models, which is to be expected as the gold standard is defined in terms of M-M links. *ALINE* produces the most accurate alignments among the 1-1 methods, with the precision and recall values that are very close to the theoretical upper bounds. Its precision is particularly impressive: on average, only one link in a thousand is not consistent with the gold standard. In terms of word accuracy, 98.97% words have no incorrect links. Out of 18,145 words, only 112 words contain incorrect links, and further 75 words could not be aligned. The ranking of the 1-1 methods is quite clear: *ALINE* followed by *IP-EM*, *1-M-EM*, *IP-align*, and BaseEM. Among the M-M methods, EM-Aggr has slightly better precision than M-M-EM, but its recall is much worse. This is probably caused by the aggregation strategy causing EM-Aggr to "lose" a significant number of correct links. In general, the entropy measure does not mirror the quality of the alignment.

Aligner	Precision	Recall	F_1 score	Entropy	L2P 1-1	L2P M-M
BaseEM	96.54	82.84	89.17	0.794	50.00	65.38
ALINE	**99.90**	**89.54**	**94.44**	0.672	**54.85**	68.74
1-M-EM	99.04	89.15	93.84	**0.636**	53.91	**69.13**
IP-align	98.30	88.49	93.14	0.706	52.66	68.25
IP-EM	99.31	89.40	94.09	0.651	53.86	68.91
M-M-EM	96.54	97.13	96.83	0.655	—	68.52
EM-Aggr	96.67	93.39	95.00	0.635	—	**69.35**
SeedMap	**97.88**	**97.44**	**97.66**	**0.634**	—	68.69
Oracle	100.0	100.0	100.0	0.640	—	69.35

Table 1: Alignment quality, entropy, and L2P conversion accuracy on the Combilex data set.

Aligner	Celex-En	CMUDict	NETtalk	OALD	Brulex
BaseEM	75.35	60.03	54.80	67.23	81.33
ALINE	**81.50**	66.46	54.90	**72.12**	**89.37**
1-M-EM	80.12	66.66	**55.00**	71.11	88.97
IP-align	78.88	62.34	53.10	70.46	83.72
IP-EM	80.95	**67.19**	54.70	71.24	87.81

Table 2: L2P word accuracy using the TiMBL-based generation system.

9 Extrinsic evaluation

In order to investigate the relationship between the alignment quality and L2P performance, we feed the alignments to two different L2P systems. The first one is a classification-based learning system employing TiMBL (Daelemans et al., 2009), which can utilize either 1-1 or 1-M alignments. The second system is the state-of-the-art online discriminative training for letter-to-phoneme conversion (Jiampojamarn et al., 2008), which accepts both 1-1 and M-M types of alignment. Jiampojamarn et al. (2008) show that the online discriminative training system outperforms a number of competitive approaches, including joint n-grams (Demberg et al., 2007), constraint satisfaction inference (Bosch and Canisius, 2006), pronunciation by analogy (Marchand and Damper, 2006), and decision trees (Black et al., 1998). The decoder module uses standard Viterbi for the 1-1 case, and a phrasal decoder (Zens and Ney, 2004) for the M-M case. We report the L2P performance in terms of word accuracy, which rewards only the completely correct output phoneme sequences. The data set is randomly split into 90% for training and 10% for testing. For all experiments, we hold out 5% of our training data to determine when to stop the online training process.

Table 1 includes the results on the Combilex data set. The two rightmost columns correspond to our two test L2P systems. We observe that although better alignment quality does not always translate into better L2P accuracy, there is nevertheless a strong correlation between the two, especially for the weaker phoneme generation system. Interestingly, *EM-Aggr* matches the L2P accuracy obtained with the gold standard alignments. However, there is no reason to claim that the gold standard alignments are optimal for the L2P generation task, so that result should not be considered as an upper bound. Finally, we note that alignment entropy seems to match the L2P accuracy better than it matches alignment quality.

Tables 2 and 3 show the L2P results on several evaluation sets: English Celex, CMUDict, NETTalk, OALD, and French Brulex. The training sizes range from 19K to 106K words. We follow exactly the same data splits as in Bisani and Ney (2008).

The TiMBL L2P generation method (Table 2) is applicable only to the 1-1 alignment models. *ALINE* produces the highest accuracy on four out of six datasets (including Combilex). The performance of *IP-EM* is comparable to *1-M-EM*, but not consistently better. *IP-align* does not seem to measure up to the other algorithms.

The discriminative approach (Table 3) is flexible enough to utilize all kinds of alignments. However, the M-M models perform clearly better than 1-1 models. The only exception is NetTalk, which

Aligner	Celex-En	CMUDict	NETTalk	OALD	Brulex
BaseEM	85.66	71.49	68.60	80.76	88.41
ALINE	87.96	75.05	69.52	81.57	94.56
1-M-EM	88.08	75.11	70.78	81.78	94.54
IP-EM	88.00	75.09	70.10	81.76	94.96
M-M-EM	88.54	75.41	70.18	82.43	95.03
EM-Aggr	**89.11**	**75.52**	**71.10**	**83.32**	**95.07**
joint n-gram	88.58	75.47	69.00	82.51	93.75

Table 3: L2P word accuracy using the online discriminative system.

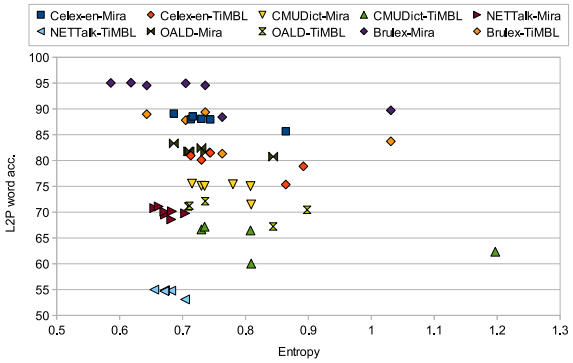


Figure 4: L2P word accuracy vs. alignment entropy.

can be attributed to the fact that NetTalk already includes double-phonemes in its original formulation. In general, the *1-M-EM* method achieves the best results among the 1-1 alignment methods, Overall, *EM-Aggr* achieves the best word accuracy in comparison to other alignment methods including the *joint n-gram* results, which are taken directly from the original paper of Bisani and Ney (2008). Except the Brulex and CMUDict data sets, the differences between *EM-Aggr* and *M-M-EM* are statistically significant according to McNemar's test at 90% confidence level.

Figure 4 contains a plot of alignment *entropy* values vs. L2P word accuracy. Each point represent an application of a particular alignment method to a different data sets. It appears that there is only weak correlation between alignment entropy and L2P accuracy. So far, we have been unable to find either direct or indirect evidence that alignment entropy is a reliable measure of letter-phoneme alignment quality.

10 Conclusion

We investigated several new methods for generating letter-phoneme alignments. The phonetic alignment is recommended for languages with little or no training data. The constraint-based approach achieves excellent accuracy at the cost of manual construction of seed mappings. The IP alignment requires no linguistic expertise and guarantees a minimal set of letter-phoneme mappings. The alignment by aggregation advances the state-of-the-art results in L2P conversion. We thoroughly evaluated the resulting alignments on several data sets by using them as input to two different L2P generation systems. Finally, we employed an independently constructed lexicon to demonstrate the close relationship between alignment quality and L2P conversion accuracy.

One open question that we would like to investigate in the future is whether L2P conversion accuracy could be improved by treating letter-phoneme alignment links as latent variables, instead of committing to a single best alignment.

Acknowledgments

This research was supported by the Alberta Ingenuity, Informatics Circle of Research Excellence (iCORE), and Natural Science of Engineering Research Council of Canada (NSERC).

References

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*, pages 77–80.

Antal Van Den Bosch and Sander Canisius. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON '06*, pages 41–49.

Walter Daelemans and Antal Van Den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in Speech Synthesis*, pages 77–89. New York, USA.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2009. *TiMBL: Tilburg Memory Based Learner, version 6.2, Reference Guide.* ILK Research Group Technical Report Series no. 09-01.

Robert I. Damper, Yannick Marchand, John DS. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 96–103, Prague, Czech Republic.

Herman Engelbrecht and Tanja Schultz. 2005. Rapid development of an afrikaans-english speech-to-speech translator. In *International Workshop of Spoken Language Translation (IWSLT)*, Pittsburgh, PA, USA.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, USA.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June. Association for Computational Linguistics.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000: 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295.

Yannick Marchand and Robert I. Damper. 2000. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219.

Yannick Marchand and Robert I. Damper. 2006. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. 2009. Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 136–144, Suntec, Singapore, August. Association for Computational Linguistics.

Korin Richmond, Robert A. J. Clark, and Sue Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *Proceedings od Interspeech*, pages 1295–1298.

Juergen Schroeter, Alistair Conkie, Ann Syrdal, Mark Beutnagel, Matthias Jilka, Volker Strom, Yeon-Jun Kim, Hong-Goo Kang, and David Kapilow. 2002. A perspective on the next challenges for TTS research. In *IEEE 2002 Workshop on Speech Synthesis*.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. In *Complex Systems*, pages 1:145–168.

Paul Taylor. 2005. Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology*.

Kristina Toutanova and Robert C. Moore. 2001. Pronunciation modeling for improved spelling correction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151, Morristown, NJ, USA.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 257–264, Boston, Massachusetts, USA.

Using Document Level Cross-Event Inference to Improve Event Extraction

Shasha Liao
New York University
715 Broadway, 7th floor
New York, NY 10003 USA
liaoss@cs.nyu.edu

Ralph Grishman
New York University
715 Broadway, 7th floor
New York, NY 10003 USA
grishman@cs.nyu.edu

Abstract

Event extraction is a particularly challenging type of information extraction (IE). Most current event extraction systems rely on local information at the phrase or sentence level. However, this local context may be insufficient to resolve ambiguities in identifying particular types of events; information from a wider scope can serve to resolve some of these ambiguities. In this paper, we use document level information to improve the performance of ACE event extraction. In contrast to previous work, we do not limit ourselves to information about events of the same type, but rather use information about other types of events to make predictions or resolve ambiguities regarding a given event. We learn such relationships from the training corpus and use them to help predict the occurrence of events and event arguments in a text. Experiments show that we can get 9.0% (absolute) gain in trigger (event) classification, and more than 8% gain for argument (role) classification in ACE event extraction.

1 Introduction

The goal of event extraction is to identify instances of a class of events in text. The ACE 2005 event extraction task involved a set of 33 generic event types and subtypes appearing frequently in the news. In addition to identifying the event itself, it also identifies all of the *participants* and *attributes* of each event; these are the *entities* that are involved in that event.

Identifying an event and its participants and attributes is quite difficult because a larger field of view is often needed to understand how facts tie together. Sometimes it is difficult even for people to classify events from isolated sentences. From the sentence:

*(1) He **left** the company.*

it is hard to tell whether it is a *Transport* event in ACE, which means that he left the place; or an *End-Position* event, which means that he retired from the company.

However, if we read the whole document, a clue like "*he planned to go shopping before he went home*" would give us confidence to tag it as a *Transport* event, while a clue like "*They held a party for his retirement*" would lead us to tag it as an *End-Position* event.

Such clues are evidence from the same event type. However, sometimes another event type is also a good predictor. For example, if we find a *Start-Position* event like "*he was named president three years ago*", we are also confident to tag (1) as *End-Position* event.

Event argument identification also shares this benefit. Consider the following two sentences:

*(2) A bomb **exploded** in Bagdad; seven people **died** while 11 were **injured**.*

*(3) A bomb **exploded** in Bagdad; the suspect got **caught** when he tried to escape.*

If we only consider the local context of the trigger "*exploded*", it is hard to determine that "*seven people*" is a likely *Target* of the *Attack* event in (2), or that the "*suspect*" is the *Attacker* of the *Attack* event, because the structures of (2) and (3) are quite similar. The only clue is from the semantic inference that a person who died may well have been a *Target* of the *Attack* event, and the person arrested is probably the *Attacker* of the *Attack* event. These may be seen as

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 789–797,
Uppsala, Sweden, 11-16 July 2010.

examples of a broader textual inference problem, and in general such knowledge is quite difficult to acquire and apply. However, in the present case we can take advantage of event extraction to learn these rules in a simpler fashion, which we present below.

Most current event extraction systems are based on phrase or sentence level extraction. Several recent studies use high-level information to aid local event extraction systems. For example, Finkel et al. (2005), Maslennikov and Chua (2007), Ji and Grishman (2008), and Patwardhan and Riloff (2007, 2009) tried to use discourse, document, or cross-document information to improve information extraction.

However, most of this research focuses on single event extraction, or focuses on high-level information within a single event type, and does not consider information acquired from other event types. We extend these approaches by introducing cross-event information to enhance the performance of multi-event-type extraction systems. Cross-event information is quite useful: first, some events co-occur frequently, while other events do not. For example, *Attack*, *Die*, and *Injure* events very frequently occur together, while *Attack* and *Marry* are less likely to co-occur. Also, typical relations among the arguments of different types of events can be helpful in predicting information to be extracted. For example, the *Victim* of a *Die* event is probably the *Target* of the *Attack* event. As a result, we extend the observation that "a document containing a certain event is likely to contain more events of the same type", and base our approach on the idea that "a document containing a certain type of event is likely to contain instances of related events". In this paper, automatically extracted within-event and cross-event information is used to aid traditional sentence level event extraction.

2 Task Description

Automatic Content Extraction (ACE) defines an event as a specific occurrence involving participants[1], and it annotates 8 types and 33 subtypes of events. We first present some ACE terminology to understand this task more easily:

- **Entity**: an object or a set of objects in one of the semantic categories of interest, referred to in the document by one or more (coreferential) entity mentions.
- **Entity mention**: a reference to an entity (typically, a noun phrase)
- **Timex:** a time expression including date, time of the day, season, year, etc.
- **Event mention**: a phrase or sentence within which an event is described, including trigger and arguments. An event mention must have one and only one trigger, and can have an arbitrary number of arguments.
- **Event trigger**: the main word that most clearly expresses an event occurrence. An ACE event trigger is generally a verb or a noun.
- **Event mention arguments (roles)**[2]: the entity mentions that are involved in an event mention, and their relation to the event. For example, event *Attack* might include participants like *Attacker, Target*, or attributes like *Time_within* and *Place*. Arguments will be taggable only when they occur within the scope of the corresponding event, typically the same sentence.

Consider the sentence:

*(4) Three **murders** occurred in **France today**, including the senseless **slaying** of **Bob Cole** and the **assassination** of **Joe Westbrook**. **Bob** was on his way home when **he** was **attacked**…*

Event extraction depends on previous phases like name identification, entity mention classification and coreference. Table 1 shows the results of this preprocessing. Note that entity mentions that share the same EntityID are coreferential and treated as the same object.

Entity(Timex) mention	head word	Entity ID	Entity type
0001-1-1	France	0001-1	GPE
0001-T1-1	Today	0001-T1	Timex
0001-2-1	Bob Cole	0001-2	PER
0001-3-1	Joe Westbrook	0001-3	PER
0001-2-2	Bob	0001-2	PER
0001-2-3	He	0001-2	PER

Table 1. An example of entities and entity mentions and their types

[1] See http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf for a description of this task.

[2] Note that we do not deal with event mention coreference in this paper, so each event mention is treated as a separate event.

There are three *Die* events, which share the same *Place* and *Time* roles, with different *Victim* roles. And there is one *Attack* event sharing the same *Place* and *Time* roles with the *Die* events.

Event type	Trigger	Role		
		Place	**Victim**	**Time**
Die	murder	0001-1-1		0001-T1-1
Die	death	0001-1-1	0001-2-1	0001-T1-1
Die	killing	0001-1-1	0001-3-1	0001-T1-1
Event type	**Trigger**	**Role**		
		Place	**Target**	**Time**
Attack	attack	0001-1-1	0001-2-3	0001-T1-1

Table2. An example of event trigger and roles

In this paper, we treat the 33 event subtypes as separate event types and do not consider the hierarchical structure among them.

3 Related Work

Almost all the current ACE event extraction systems focus on processing one sentence at a time (Grishman et al., 2005; Ahn, 2006; Hardy et al. 2006). However, there have been several studies using high-level information from a wider scope:

Maslennikov and Chua (2007) use discourse trees and local syntactic dependencies in a pattern-based framework to incorporate wider context to refine the performance of relation extraction. They claimed that discourse information could filter noisy dependency paths as well as increasing the reliability of dependency path extraction.

Finkel et al. (2005) used Gibbs sampling, a simple Monte Carlo method used to perform approximate inference in factored probabilistic models. By using simulated annealing in place of Viterbi decoding in sequence models such as HMMs, CMMs, and CRFs, it is possible to incorporate non-local structure while preserving tractable inference. They used this technique to augment an information extraction system with long-distance dependency models, enforcing label consistency and extraction template consistency constraints.

Ji and Grishman (2008) were inspired from the hypothesis of "One Sense Per Discourse" (Yarowsky, 1995); they extended the scope from a single document to a cluster of topic-related documents and employed a rule-based approach to propagate consistent trigger classification and event arguments across sentences and documents. Combining global evidence from related documents with local decisions, they obtained an appreciable improvement in both event and event argument identification.

Patwardhan and Riloff (2009) proposed an event extraction model which consists of two components: a model for sentential event recognition, which offers a probabilistic assessment of whether a sentence is discussing a domain-relevant event; and a model for recognizing plausible role fillers, which identifies phrases as role fillers based upon the assumption that the surrounding context is discussing a relevant event. This unified probabilistic model allows the two components to jointly make decisions based upon both the local evidence surrounding each phrase and the "peripheral vision".

Gupta and Ji (2009) used cross-event information within ACE extraction, but only for recovering implicit time information for events.

4 Motivation

We analyzed the sentence-level baseline event extraction, and found that many events are missing or spuriously tagged because the local information is not sufficient to make a confident decision. In some local contexts, it is easy to identify an event; in others, it is hard to do so. Thus, if we first tag the easier cases, and use such knowledge to help tag the harder cases, we might get better overall performance. In addition, global information can make the event tagging more consistent at the document level.

Here are some examples. For trigger classification:

*The pro-reform director of Iran's biggest-selling daily newspaper and official organ of Tehran's municipality has **stepped** down following the **appointment** of a conservative …it was **founded** a decade ago … but a conservative city council was **elected** in the February 28 municipal polls … Mahmud Ahmadi-Nejad, reported to be a hardliner among conservatives, was **appointed** mayor on Saturday …**Founded** by former mayor Gholamhossein Karbaschi, Hamshahri…*

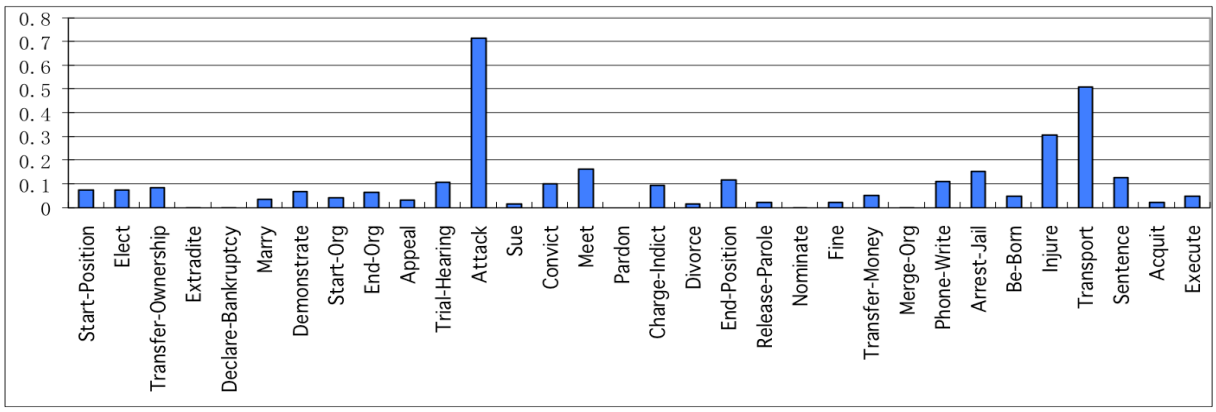


Figure 1. Conditional probability of the other 32 event types in documents where a *Die* event appears

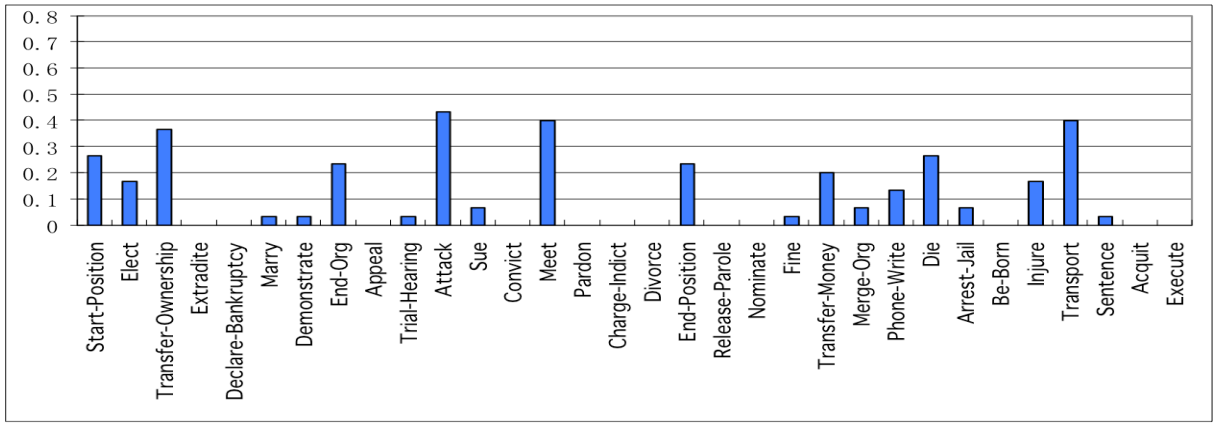


Figure 2. Conditional probability of the other 32 event types in documents where a *Start-Org* event appears

The sentence level baseline system finds event triggers like "*founded*" (trigger of *Start-Org*), "*elected*" (trigger of *Elect*), and "*appointment*" (trigger of *Start-Position*), which are easier to identify because these triggers have more specific meanings. However, it does not recognize the trigger "*stepped*" (trigger of *End-Position*) because in the training corpus "*stepped*" does not always appear as an *End-Position* event, and local context does not provide enough information for the MaxEnt model to tag it as a trigger. However, in the document that contains related events like *Start-Position*, "*stepped*" is more likely to be tagged as an *End-Position* event.

For argument classification, the cross-event evidence from the document level is also useful:

*British officials say they believe **Hassan** was a blindfolded woman seen being **shot** in the head by a hooded militant on a video obtained but not aired by the Arab television station Al-Jazeera. **She** would be the first foreign woman to **die** in the wave of kidnappings in Iraq…**she**'s been **killed** by (men in pajamas), turn Iraq upside down and find them.*

From this document, the local information is not enough for our system to tag "*Hassan*" as the target of an *Attack* event, because it is quite far from the trigger "*shot*" and the syntax is somewhat complex. However, it is easy to tag "*she*" as the *Victim* of a *Die* event, because it is the object of the trigger "*killed*". As "*she*" and "*Hassan*" are co-referred, we can use this easily tagged argument to help identify the harder one.

4.1 Trigger Consistency and Distribution

Within a document, there is a strong trigger consistency: if one instance of a word triggers an event, other instances of the same word will trigger events of the same type[3].

There are also strong correlations among event types in a document. To see this we calculated the conditional probability (in the ACE corpus) of a certain event type appearing in a document when another event type appears in the same document.

[3] This is true over 99.4% of the time in the ACE corpus.

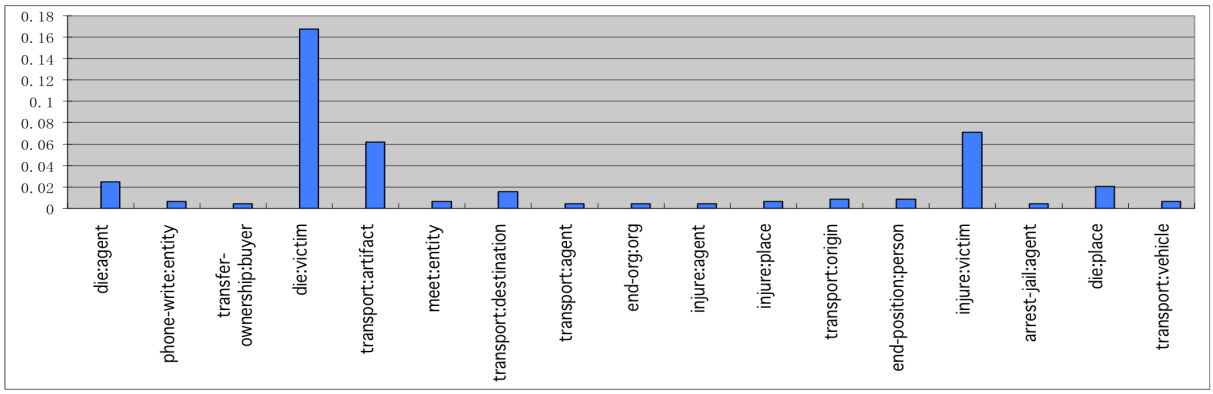


Figure 3. Conditional probability of all possible roles in other event types for entities that are the *Target*s of *Attack* events (roles with conditional probability below 0.002 are omitted)

Event	Cond. Prob.
Attack	0.714
Transport	0.507
Injure	0.306
Meet	0.164
Arrest-Jail	0.153
Sentence	0.126
Phone-Write	0.111
End-Position	0.116
Trial-Hearing	0.105
Convict	0.100

Table 3. Events co-occurring with *die* events with conditional probability > 10%

As there are 33 subtypes, there are potentially 33·32/2=528 event pairs. However, only a few of these appear with substantial frequency. For example, there are only 10 other event types that occur in more than 10% of the documents in which a *die* event appears. From Table 3, we can see that *Attack*, *Transport* and *Injure* events appear frequently with *Die*. We call these the related event types for *Die* (see Figure 1 and Table 3).

The same thing happens for *Start-Org* events, although its distribution is quite different from *Die* events. For *Start-Org*, there are more related events like *End-Org*, *Start-Position*, and *End-Position* (Figure 2). But there are 12 other event types which never appear in documents containing *Start-Org* events.

From the above, we can see that the distributions of different event types are quite different, and these distributions might be good predictors for event extraction.

4.2 Role Consistency and Distribution

Normally one entity, if it appears as an argument of multiple events *of the same type* in a single document, is assigned the same role each time.[4]

There is also a strong relationship between the roles when an entity participates in different types of events in a single document. For example, we checked all the entities in the ACE corpus that appear as the *Target* role for an *Attack* event, and recorded the roles they were assigned for other event types. Only 31 other event-role combinations appeared in total (out of 237 possible with ACE annotation), and 3 clearly dominated. In Figure 3, we can see that the most likely roles for the *Target* role of the *Attack* event are the *Victim* role of the *Die* or *Injure* event and the *Artifact* role of the *Transport* event. The last of these corresponds to troop movements prior to or in response to attacks.

5 Cross-event Approach

In this section we present our approach to using document-level event and role information to improve sentence-level ACE event extraction.

Our event extraction system is a two-pass system where the sentence-level system is first applied to make decisions based on local information. Then the confident local information is collected and gives an approximate view of the content of the document. The document level system is finally applied to deal with the cases which the local

[4] This is true over 97% of the time in the ACE corpus.

system can't handle, and achieve document consistency.

5.1 Sentence-level Baseline System

We use a state-of-the-art English IE system as our baseline (Grishman et al. 2005). This system extracts events independently for each sentence, because the definition of event mention argument constrains them to appear in the same sentence. The system combines pattern matching with statistical models. In the training process, for every event mention in the ACE training corpus, patterns are constructed based on the sequences of constituent heads separating the trigger and arguments. A set of Maximum Entropy based classifiers are also trained:

- **Argument Classifier:** to distinguish arguments of a potential trigger from non-arguments;
- **Role Classifier:** to classify arguments by argument role.
- **Reportable-Event Classifier (Trigger Classifier):** Given a potential trigger, an event type, and a set of arguments, to determine whether there is a reportable event mention.

In the test procedure, each document is scanned for instances of triggers from the training corpus. When an instance is found, the system tries to match the environment of the trigger against the set of patterns associated with that trigger. This pattern-matching process, if successful, will assign some of the mentions in the sentence as arguments of a potential event mention. The argument classifier is applied to the remaining mentions in the sentence; for any argument passing that classifier, the role classifier is used to assign a role to it. Finally, once all arguments have been assigned, the reportable-event classifier is applied to the potential event mention; if the result is successful, this event mention is reported.[5]

5.2 Document-level Confident Information Collector

To use document-level information, we need to collect information based on the sentence-level baseline system. As it is a statistically-based model, it can provide a value that indicates how likely it is that this word is a trigger, or that the mention is an argument and has a particular role. We want to see if this value can be trusted as a confidence score. To this end, we set different thresholds from 0.1 to 1.0 in the baseline system output, and only evaluate triggers, arguments or roles whose confidence score is above the threshold. Results show that as the threshold is raised, the precision generally increases and the recall falls. This indicates that the value is consistent and a useful indicator of event/argument confidence (see Figure 4).[6]

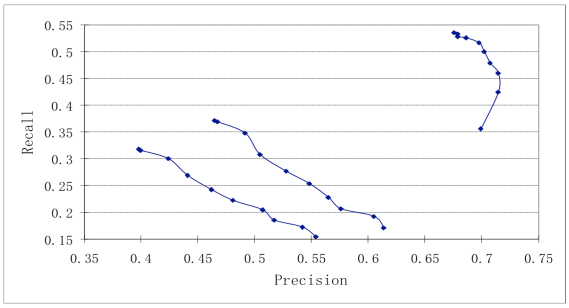


Figure 4. The performance of different confidence thresholds in the baseline system on the development set

To acquire confident document-level information, we only collect triggers and roles tagged with high confidence. Thus, a trigger threshold *t_threshold* and role threshold *r_threshold* are set to remove low confidence triggers and arguments. Finally, a table with *confident event* information is built. For every event, we collect its trigger and event type; for every argument, we use co-reference information and record every entity and its role(s) in events of a certain type.

To achieve document consistency, in cases where the baseline system assigns a word to triggers for more than one event type, if the margin between the probability of the highest and the second highest scores is above a threshold *m_threshold*, we only keep the event type with highest score and record this in the *confident-event* table. Otherwise (if the margin is smaller) the event type assignments will be recorded in a separate *conflict* table. The same strategy is applied to argument/role conflicts. We will not use information in the conflict table to infer the event type or argument/roles for other event mentions, because we cannot

[5] If the event arguments include some assigned by the pattern-matching process, the event mention is accepted unconditionally, bypassing the reportable- event classifier.

[6] The trigger classification curve doesn't follow the expected recall/precision trade-off, particularly at high thresholds. This is due, at least in part, to the fact that some events bypass the reportable-event classifier (trigger classifier) (see footnote 5). At high thresholds this is true of the bulk of the events.

confidently resolve the conflict. However, the event type and argument/role assignments in the conflict table will be included in the final output because the local confidence for the individual assignments is high.

As a result, we finally build two document-level confident-event tables: the event type table and the argument (role) table. A conflict table is also built but not used for further predictions (see Table 4).

Confident table		
Event type table		
Trigger	**Event Type**	
Met	Meet	
Exploded	Attack	
Went	Transport	
Injured	Injure	
Attacked	Attack	
Died	Die	
Argument role table		
Entity ID	**Event type**	**Role**
0004-T2	Die	Time Within
0004-6	Die	Place
0004-4	Die	Victim
0004-7	Die	Agent
0004-11	Attack	Target
0004-T3	Attack	Time Within
0004-12	Attack	Place
0004-10	Attack	Attacker
Conflict table		
Entity ID	**Event type**	**Roles**
0004-8	Attack	Victim, Agent

Table 4. Example of document-level confident-event table (event type and argument role entries) and conflict table

5.3 Statistical Cross-event Classifiers

To take advantage of cross-event relationships, we train two additional MaxEnt classifiers – a document-level trigger and argument classifier – and then use these classifiers to infer additional events and event arguments. In analyzing new text, the trigger classifier is first applied to tag an event, and then the argument (role) classifier is applied to tag possible arguments and roles of this event.

5.3.1 Document Level Trigger Classifier

From the document-level confident-event table, we have a rough view of what kinds of events are reported in this document. The trigger classifier predicts whether a word is the trigger of an event, and if so of what type, given the information (from the confident-event table) about other types of events in the document. Each feature of this classifier is the conjunction of:

- The base form of the word
- An event type
- A binary indicator of whether this event type is present elsewhere in the document

(There are 33 event types and so 33 features for each word).

5.3.2 Document Level Argument (Role) Classifier

The role classifier predicts whether a given mention is an argument of a given event and, if so, what role it takes on, again using information from the confident-event table about other events.

As noted above, we assume that the role of an entity is unique for a specific event type, although an entity can take on different roles for different event types. Thus, if there is a conflict in the document level table, the collector will only keep the one with highest confidence, or discard them all. As a result, every entity is assigned a unique role with respect to a particular event type, or *null* if it is not an argument of a certain event type.

Each feature is the conjunction of:

- The event type we are trying to assign an argument/role to.
- One of the 32 other event types
- The role of this entity with respect to the other event type elsewhere in the document, or *null* if this entity is not an argument of that type of event

5.4 Document Level Event Tagging

At this point, the low-confidence triggers and arguments (roles) have been removed and the document-level confident-event table has been built; the new classifiers are now used to *augment* the confident tags that were previously assigned based on local information.

For trigger tagging, we only apply the classifier to the words that do not have a confident local labeling; if the trigger is already in the document level confident-event table, we will not re-tag it.

performance / system/human	Trigger classification			Argument classification			Role classification		
	P	R	F	P	R	F	P	R	F
Sentence-level baseline system	67.56	53.54	59.74	46.45	37.15	41.29	41.02	32.81	36.46
Within-event-type rules	63.03	59.90	61.43	48.59	46.16	47.35	43.33	41.16	42.21
Cross-event statistical model	68.71	68.87	**68.79**	50.85	49.72	**50.28**	45.06	44.05	**44.55**
Human annotation1	59.2	59.4	59.3	60.0	69.4	64.4	51.6	59.5	55.3
Human annotation2	69.2	75.0	72.0	62.7	85.4	72.3	54.1	73.7	62.4

Table 5. Overall performance on blind test data

The argument/role tagger is then applied to all events—those in the confident-event table and those newly tagged. For argument tagging, we only consider the entity mentions in the same sentence as the trigger word, because by the ACE event guidelines, the arguments of an event should appear within the same sentence as the trigger. For a given event, we re-tag the entity mentions that have not already been assigned as arguments of that event by the confident-event or conflict table.

6 Experiments

We followed Ji and Grishman (2008)'s evaluation and randomly select 10 newswire texts from the ACE 2005 training corpora as our development set, which is used for parameter tuning, and then conduct a blind test on a separate set of 40 ACE 2005 newswire texts. We use the rest of the ACE training corpus (549 documents) as training data for both the sentence-level baseline event tagger and document-level event tagger.

To compare with previous work on within-event propagation, we reproduced Ji and Grishman (2008)'s approach for cross-sentence, within-event-type inference (see "within-event-type rules" in Table 5). We applied their within-document inference rules using the cross-sentence confident-event information. These rules basically serve to adjust trigger and argument classification to achieve document-wide consistency. This process treats each event type separately: information about events of a given type is used to infer information about other events of the same type.

We report the overall Precision (P), Recall (R), and F-Measure (F) on blind test data. In addition, we also report the performance of two human annotators on 28 ACE newswire texts (a subset of the blind test set).[7]

From the results presented in Table 5, we can see that using the document level cross-event information, we can improve the F score for trigger classification by 9.0%, argument classification by 9.0%, and role classification by 8.1%. Recall improved sharply, demonstrating that cross-event information could recover information that is difficult for the sentence-level baseline to extract; precision also improved over the baseline, although not as markedly.

Compared to the within-event-type rules, the cross-event model yields much more improvement for trigger classification: rule-based propagation gains 1.7% improvement while the cross-event model achieves a further 7.3% improvement. For argument and role classification, the cross-event model also gains 3% and 2.3% above that obtained by the rule-based propagation process.

7 Conclusion and Future Work

We propose a document-level statistical model for event trigger and argument (role) classification to achieve document level within-event and cross-event consistency. Experiments show that document-level information can improve the performance of a sentence-level baseline event extraction system.

The model presented here is a simple two-stage recognition process; nonetheless, it has proven sufficient to yield substantial improvements in event recognition and event

[7] The final key was produced by review and adjudication of the two annotations by a third annotator, which indicates that the event extraction task is quite difficult and human agreement is not very high.

argument recognition. Richer models, such as those based on joint inference, may produce even greater gains. In addition, extending the approach to cross-document information, following (Ji and Grishman 2008), may be able to further improve performance.

References

David Ahn. 2006. The stages of event extraction. In *Proc. COLING/ACL 2006 Workshop on Annotating and Reasoning about Time and Events.* Sydney, Australia.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proc. 43rd Annual Meeting of the Association for Computational Linguistics,* pages 363–370, Ann Arbor, MI, June.

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. In *Proc. ACE 2005 Evaluation Workshop*, Gaithersburg, MD.

Prashant Gupta, Heng Ji. 2009. Predicting Unknown Time Arguments based on Cross-Event Propagation. In *Proc. ACL-IJCNLP 2009.*

Hilda Hardy, Vika Kanchakouskaya and Tomek Strzalkowski. 2006. Automatic Event Classification Using Surface Text Features. In *Proc. AAAI06 Workshop on Event Extraction and Synthesis.* Boston, MA.

H. Ji and R. Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In *Proc. ACL-08: HLT,* pages 254–262, Columbus, OH, June.

M. Maslennikov and T. Chua. 2007. A Multi resolution Framework for Information Extraction from Free Text. In *Proc. 45th Annual Meeting of the Association of Computational Linguistics,* pages 592–599, Prague, Czech Republic, June.

S. Patwardhan and E. Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In *Proc. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007,* pages 717–727, Prague, Czech Republic, June.

Patwardhan, S. and Riloff, E. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In *Proc. Conference on Empirical Methods in Natural Language Processing 2009, (EMNLP-09).*

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proc. ACL 1995.* Cambridge, MA.

Now, where was I?
Resumption strategies for an in-vehicle dialogue system

Jessica Villing
Graduate School of Language Technology and
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
jessica@ling.gu.se

Abstract

In-vehicle dialogue systems often contain more than one application, e.g. a navigation and a telephone application. This means that the user might, for example, interrupt the interaction with the telephone application to ask for directions from the navigation application, and then resume the dialogue with the telephone application. In this paper we present an analysis of interruption and resumption behaviour in human-human in-vehicle dialogues and also propose some implications for resumption strategies in an in-vehicle dialogue system.

1 Introduction

Making it useful and enjoyable to use a dialogue system is always important. The dialogue should be easy and intuitive, otherwise the user will not find it worth the effort and instead prefer to use manual controls or to speak to a human.

However, when designing an in-vehicle dialogue system there is one more thing that needs to be taken into consideration, namely the fact that the user is performing an additional, safety critical, task - driving. The so-called 100-car study (Neale et al., 2005) revealed that secondary task distraction is the largest cause of driver inattention, and that the handling of wireless devices is the most common secondary task. Even if spoken dialogue systems enables manouvering of devices without using hands or eyes, it is crucial to adjust the interaction to the in-vehicle environment in order to minimize distraction from the interaction itself. Therefore the dialogue system should consider the cognitive load of the driver and adjust the dialogue accordingly. One way of doing this is to continously measure the cognitive workload level of the driver and, if the workload is high, determine type of workload and act accordingly. If the workload is *dialogue-induced* (i.e. caused by the dialogue itself), it might be necessary to rephrase or offer the user help with the task. If the workload is *driving-induced* (i.e. caused by the driving task), the user might need information that is crucial for the driving task (e.g. get navigation instructions), or to pause the dialogue in order to enable the user to concentrate on the driving task (Villing, 2009). Both the driver and the system should be able to initiate interruptions.

When the interaction with a dialogue system has been interrupted, e.g. because the user has not answered a question, it is common that the system returns to the top menu. This means that if the user wants to finish the interrupted task she has to restart from the beginning, which is both time-consuming and annoying. Instead, the dialogue system should be able to either pause until the workload is low or change topic and/or domain, and then resume where the interruption took place. However, resumption of an interrupted topic needs to be done in a way that minimizes the risk that the cognitive workload increases again. Although a lot of research has been done regarding dialogue system output, very little work has been done regarding resumption of an interrupted topic. In this paper we will analyse human-human in-vehicle dialogue to find out how resumptions are done in human-human dialogue and propose some implications for resumption strategies in a dialogue system.

2 Related work

To study resumption behaviour, Yang (2009), carried out a data collection where the participants were switching between an ongoing task (a card game) and a real-time task (a picture game). The participants randomly had to interrupt the ongoing task to solve a problem in the real-time task. When studying the resumption behaviour after an

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 798–805,
Uppsala, Sweden, 11-16 July 2010.

interruption to the real-time task they found that the resuming utterance contained various amounts and types of redundant information depending on whether the interruption occured in the middle of a card discussion, at the end of a card or at the end of a card game. If the interruption occured in the middle of a card discussion it was possible to make a distinction between *utterance restatement* (repeat one's own utterance, repeat the dialogue partners utterance or clarification of the dialogue partners utterance) and *card review* (reviewing all the cards on hand although this information had already been given). They found that the behaviour is similar to grounding behaviour, where the speaker use repetition and requests for repetition to ensure that the utterance is understood.

3 Data collection

A data collection has been carried out within the DICO project (see, for example, (Larsson and Villing, 2007)) to study how an additional distraction or increase in the cognitive load would affect a driver's dialogue behaviour. The goal was to elicit a natural dialogue (as opposed to giving the driver a constructed task such as for example a math task) and make the participants engage in the conversation.

The participants (two female and six male) between the ages of 25 and 36 drove a car in pairs while interviewing each other. The interview questions and the driving instructions were given to the passenger, hence the driver knew neither what questions to discuss nor the route in advance. Therefore, the driver had to signal, implicitly or explicitly, when she wanted driving instructions and when she wanted a new question to discuss. The passenger too had to have a strategy for when to change topic. The reasons for this setup was to elicit a natural and fairly intense dialogue and to force the participants to frequently change topic and/or domain (e.g. to get driving instructions). The participants changed roles after 30 minutes, which meant that each participant acted both as driver and as passenger. The cognitive load of the driver was measured in two ways. The driver performed a Tactile Detection Task (TDT) (van Winsum et al., 1999). When using a TDT, a buzzer is attached to the driver's wrist. The driver is told to push a button each time the summer is activated. Cognitive load is determined by measuring hit-rate and reaction time. Although the TDT task in itself might cause an increased workload level, the task is performed during the whole session and thereby it is possible to distinguish high workload caused by something else but the TDT task.

Workload was also measured by using an IDIS system (Broström et al., 2006). IDIS determines workload based on the driver's behaviour (for example, steering wheel movements or applying the brake). What differs between the two measurements is that the TDT measures the actual workload of each driver, while IDIS makes its assumptions based on knowledge of what manouvres are usually cognitively demanding.

The participants were audio- and videotaped, the recordings are transcribed with the transcription tool ELAN[1], using an orthographic transcription. All in all 3590 driver utterances and 4382 passenger utterances are transcribed. An annotation scheme was designed to enable analysis of utterances with respect to topic change for each domain.

Domain and topic was defined as:

- *interview* domain: discussions about the interview questions where each interview question was defined as a topic
- *navigation* domain: navigation-related discussions where each navigation instruction was defined as a topic
- *traffic* domain: discussions about the traffic situation and fellow road-users where each comment not belonging to a previous event was defined as a topic
- *other* domain: anything that does not fit within the above domains where each comment not belonging to a previous event was defined as a topic

Topic changes has been coded as follows:

- *begin-topic*: whatever → new topic
  - I.e., the participants start discussing an interview question, a navigation instruction, make a remark about the traffic or anything else that has not been discussed before.
- *end-topic*: finished topic → whatever

[1] http://www.lat-mpi.eu/tools/elan/

  - A topic is considered finished if a question is answered or if an instruction or a remark is confirmed.

- *interrupt-topic*: unfinished topic → whatever
  - An utterance is considered to interrupt if it belongs to another topic than the previous utterance and the previous topic has not been ended with an *end-topic*.
- *resume-topic*: whatever → unfinished topic
  - A topic is considered to be resumed if it has been discussed earlier but was not been finished by an *end-topic* but instead interrupted with an *interrupt-topic*.
- *reraise-topic*: whatever → finished topic
  - A topic is considered to be reraised if it has been discussed before and then been finished with an *end-topic*.

The utterances have been categorised according to the following schema:

- DEC: declarative
  - (*"You are a Leo and I am a Gemini"*, *"This is Ekelund Street"*)
- INT: interrogative
  - (*"What do you eat for breakfast?"*, *"Should we go back after this?"*)
- IMP: imperative
  - (*"Go on!"*)
- ANS: *"yes"* or *"no"* answer (and variations such as *"sure, absolutely, nope, no way"*)
- NP: bare noun phrase
  - (*"Wolfmother"*, *"Otterhall Street"*)
- ADVP: bare adverbial phrase
  - (*"Further into Karlavagn Street"*)
- INC: incomplete phrase
  - (*"Well, did I answer the"*, *"Should we"*)

Cognitive load has been annotated as:

- **reliable workload**: annotated when workload is reliably high according to the TDT (reliability was low if response button was pressed more than 2 times after the event).
- **high**: high workload according to IDIS
- **low**: low workload according to IDIS

The annotation schema has not been tested for inter-coder reliability. While full reliability testing would have further strengthened the results, we believe that our results are still useful as a basis for future implementation and experimental work.

4 Results

The codings from the DICO data collection has been analysed with respect to interruption and resumption of topics (*interrupt-topic* and *resume-topic*, respectively). Interruption can be done in two ways, either to pause the dialogue or to change topic and/or domain. In the DICO corpus there are very few interruptions followed by a pause. The reason is probably that both the driver and the passenger were strongly engaged in the interview and navigation tasks. The fact that the driver did not know the route elicited frequent switches to the navigation domain done by both the driver and the passenger, as can be seen in Figure 1. Therefore, we have only analysed interruption and resumption from and to the interview and navigation domains.

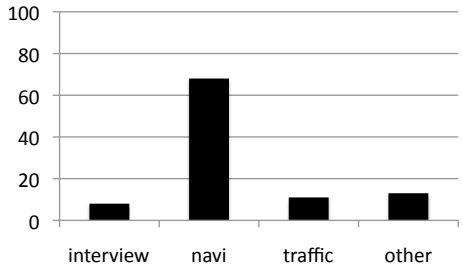


Figure 1: Distribution of utterances coded as *interrupt-topic* for each domain, when interrupting from an interview topic.

4.1 Redundancy

The easiest way of resuming an interrupted topic in a dialogue system is to repeat the last phrase that was uttered before the interruption. One disdavantage of this method is that the dialogue system might be seen as tedious, especially if there are several interruptions during the interaction. We wanted to see if the resuming utterances in human-human dialogue are redundant and if redundancy has anything to do with the length of the interruption. We therefore sorted all utterances coded

as *resume-topic* in two categories, those which contained redundant information when comparing with the last utterance before the interruption, and those which did not contain and redundant information. As a redundant utterance we counted all utterances that repeated one or more words from the last utterance before the interruption. We then counted the number of turns between the interruption and resumption. The number of turns varied between 1 and 42. The result can be seen in Figure 2.

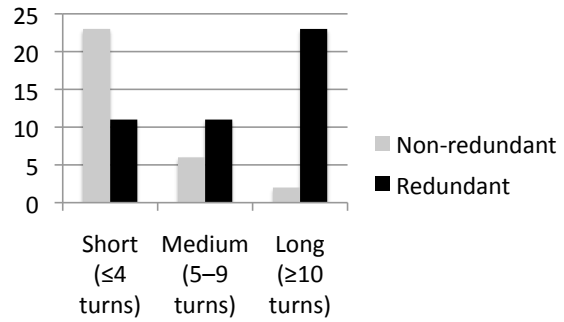


Figure 2: Number of redundant utterances depending on length of interruption.

As can be seen, there are twice as many non-redundant as redundant utterances after a short interruption ($\leq$4 turns), while there are almost solely redundant utterances after a long interruption ($\geq$10 turns). The average number of turns is 3,5 when no redundancy occur, and 11,5 when there are redundancy. When the number of turns exceeds 12, there are only redundant utterances.

4.2 Category

Figure 3 shows the distribution, sorted per category, of driver utterances when resuming to an interview and a navigation topic. Figure 4 shows the corresponding figures for passenger utterances.

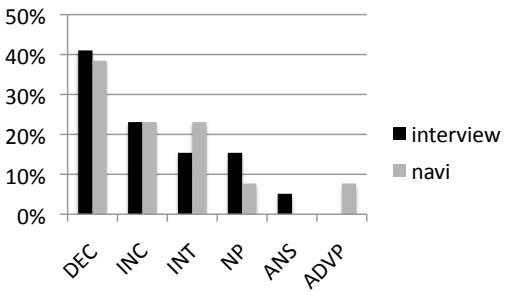


Figure 3: Driver resuming to the interview and navigation domains.

The driver's behaviour is similar both when resuming to an interview and a navigation topic. Declarative phrases are most common, followed by incomplete, interrogative (for interview topics) and noun phrases.

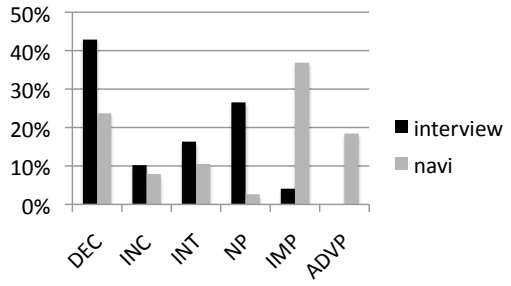


Figure 4: Passenger resuming to the interview and navigation domains.

When looking at the passenger utterances we see a lot of variation between the domains. When resuming to an interview topic the passenger uses mostly declarative phrases, followed by noun phrases and interrogative phrases. When resuming to a navigation topic imperative phrases are most common, followed by declarative phrases. Only the passenger use imperative phrases, probably since the passenger is managing both the interview questions and the navigation instructions and therefore is the one that is forcing both the interview and the navigation task through.

4.3 Workload level

The in-vehicle environment is forcing the driver to carry out tasks during high cognitive workload. To minimize the risk of increasing the workload further, an in-vehicle dialogue system should be able to decide when to interrupt and when to resume a topic depending on the driver's workload level.

The figures in this section shows workload level and type of workload during interruption and resumption to and from topics in the interview domain. When designing the interview and navigation tasks that were to be carried out during the data collection, we focused on designing them so that the participants were encouraged to discuss as much as possible with each other. Therefore, the navigation instructions sometimes were hard to understand, which forced the participants to discuss the instructions and together try to interpret them. Therefore we have not analysed the workload level while interrupting and resuming topics in the navigation domain since the result might be

misleading.

Type of workload is determined by analysing the TDT and IDIS signals described in 3. Workload is considered to be dialogue-induced when only the TDT is indicating high workload (since the TDT indicates that the driver is carrying out a task that is cognitively demanding but IDIS is not indicating that the driving task is demanding at the moment), driving-induced when both the TDT and IDIS is indicating high workload (since the TDT is indicating that the workload level is high and IDIS is indicating that the driving task is demanding) and possibly driving-induced when only IDIS is indicating high workload (since IDIS admittedly is indicating that the driving task is demanding but the TDT indicates that the driver's workload is low, it could then be that this particular driver does not experience the driving task demanding even though the average driver does) (Villing, 2009).

The data has been normalized for variation in workload time. The diagrams shows the distribution of interruption and resumption utterances made by the driver and the passenger, respectively.

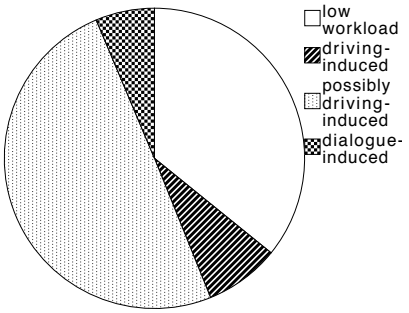


Figure 5: Workload while the driver is interrupting an interview topic.

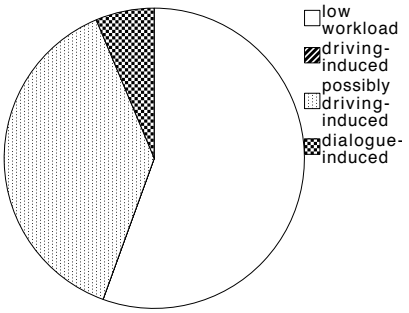


Figure 6: Workload while the passenger is interrupting an interview topic.

Figures 5 and 6 show driver workload level while the driver and the passenger (respectively) are interrupting from the interview domain. The driver most often interrupts during a possible driving-induced or low workload, the same goes for the passenger but in opposite order. It is least common for the driver to interrupt during dialogue- or driving-induced workload, while the passenger rarely interrupts during dialogue-induced and never during driving-induced workload.

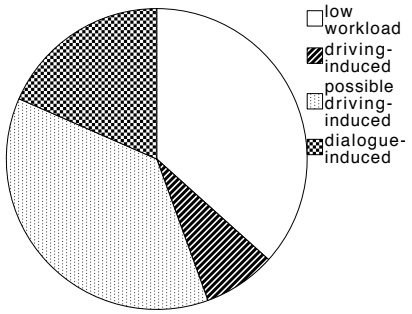


Figure 7: Workload while driver is resuming to the interview domain.

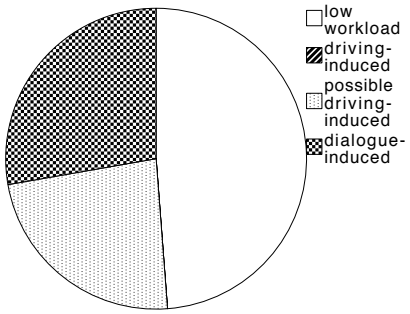


Figure 8: Workload while passenger is resuming to the interview domain.

Figures 7 and 8 show workload level while the driver and the passenger (respectively) are resuming to the interview domain. The driver most often resumes while the workload is low or possibly driving-induced, while the passenger is mostly resuming during low workload and never during driving-induced workload.

5 Discussion

For both driver and passenger, the most common way to resume an interview topic is to use a declarative utterance, which is illustrated in Figure 3. When studying the utterances in detail we can see that there is a difference when comparing information redundancy similar to what Yang (2009) describe in their paper. They compared grade of

redundancy based on where in the dialogue the interruption occur, what we have looked at in the DICO corpus is how many turns the interrupting discussion contains.

As Figure 2 shows, if the number of turns is about three (on average, 3,5), the participants tend to continue the interrupted topic exactly where it was interrupted, without considering that there had been any interruption. The speaker however often makes some sort of sequencing move to announce that he or she is about to switch domain and/or topic, either by using a standard phrase or by making an extra-lingustic sound like, for example, lipsmack or breathing (Villing et al., 2008). Example (1) shows how the driver interrupts a discussion about what book he is currently reading to get navigation instructions:

(1) Driver: What I read now is Sofie's world.

Driver (*interrupting*): Yes, where do you want me to drive?

Passenger: Straight ahead, straight ahead.

Driver: Straight ahead. Alright, I'll do that.

Passenger (*resuming*): Alright [*sequencing move*]. Enemy of the enemy was the last one I read. [*DEC*]

If the number of turns is higher than ten (on average, 11,5) the resuming speaker makes a redundant utterance, repeating one or more words from the last utterance before the interruption. See example (2):

(2) Driver: Actually, I have always been interested in computers and technology.

Passenger (*interrupting*): Turn right to Vasaplatsen. Is it here? No, this is Grönsakstorget.

Driver: This is Grönsakstorget. We have passed Vasaplatsen.

. . . (*Discussion about how to turn around and get back to Vasaplatsen, all in all 21 turns.*)

Driver (*resuming*): Well, as I said [*sequencing move*]. I have always been interested in computer and computers and technology and stuff like that. [*DEC*]

The passenger often uses a bare noun phrase to resume, the noun phrase can repeat a part of the interview question. For example, after a discussion about wonders of the world, which was interrupted by a discussion about which way to go next, the passenger resumed by uttering the single word "wonders" which was immediatly understood by the driver as a resumption to the interview topic. The noun phrase can also be a key phrase in the dialogue partner's answer as in example (3) where the participants discuss their favourite band:

(3) Driver: I like Wolfmother, do you know about them?

Passenger: I've never heard about them. [...] You have to bring a cd so I can listen to them.

Driver (*interrupting*): Where was I supposed to turn?

. . . (*Navigation discussion, all in all 13 turns.*)

Passenger (*resuming*): [*LAUGHS*]Wolfmother. [*NP*]

When resuming to the navigation domain, the driver mostly uses a declarative phrase, typically to clarify an instruction. It is also common to use an interrogative phrase or an incomplete phrase such as *"should I..."* which the passenger answers by clarifying which way to go. The passenger instead uses mostly imperative phrases as a reminder of the last instruction, such as *"keep straight on"*.

When the speakers interrupts an interview topic they mostly switch to the navigation domain, see Figure 1. That means that the most common reason for the speaker to interrupt is to ask for or give information that is crucial for the driving task (as opposed for the other and traffic domains, which are mostly used to signal that the speaker's cognitive load level is high (Villing et al., 2008)). As can be seen in Figures 5 and 6, the driver mostly interrupts the interview domain during a possible driving-induced workload while the passenger mostly interrupts during low workload. As noted above (see also Figure 3), the utterances are mostly declarative (*"this is Ekelund Street"*), interrogative (*"and now I turn left?"*) or incomplete (*"and then..."*), while the passenger gives additional information that the driver has not asked for explicitly but the passenger judges that the driver might need (*"just go straight ahead in the next crossing"*, *"here is where we should turn towards Järntorget"*). Hence, it seems like the driver interrupts to make clarification utterances that must be answered immediately, for example, right before a

crossing when the driver has pressed the brakes or turned on the turn signal (and therefore the IDIS system signals high workload which is interpreted as driving-induced workload) while the passenger take the chance to give additional information in advance, before it is needed, and the workload therefore is low.

Figure 7 shows that the driver mostly resumes to the interview domain during low or possible driving-induced workload. Since the IDIS system makes its assumption on driving behaviour, based on what the average driver finds cognitively demanding, it might sometimes be so that the system overgenerates and indicates high workload even though the driver at hand does not find the driving task cognitively demanding. This might be an explanation to these results, since the driver often resumes to an interview topic although he or she is, for example, driving through a roundabout or pushing the brakes. It is also rather common that the driver is resuming to an interview question during dialogue-induced workload, perhaps because she has started thinking about an answer to a question and therefore the TDT indicates high workload and the IDIS does not. The passenger mostly resumes to the interview domain during low workload, which indicates that the passenger analyses both the traffic situation and the state of mind of the driver before he or she wants to draw the drivers attention from the driving task.

6 Implications for in-vehicle dialogue systems

In this paper we point at some of the dialogue strategies that are used in human-human dialogue during high cognitive load when resuming to an interrupted topic. These strategies should be taken under consideration when implementing an in-vehicle dialogue system.

To make the dialogue natural and easy to understand the dialogue manager should consider which domain it will resume to and the number of turns between the interruption and resumption before deciding what phrase to use as output. For example, the results indicate that it might be more suitable to use a declarative phrase when resuming to a domain where the system is asking the user for information, for example when adding songs to a play list at the mp3-player (cf. the interview domain). If the number of turns are 4 or less, it probably does not have to make a redundant utterance at all, but may continue the discussion where it was interrupted. If the number of turns exceeds 4 it is probably smoother to let the system just repeat one or more keywords from the interrupted utterance to make the user understand what topic should be discussed, instead of repeating the whole utterance or even start the task from the beginning. This will make the system feel less tedious which should have a positive effect on the cognitive workload level. However, user tests are probably needed to decide how much redundant information is necessary when talking to a dialogue system, since it may well differ from talking to a human being who is able to help the listener understand by, for example, emphasizing certain words in a way that is currently impossible for a computer. When resuming to a domain where the system has information to give to the user it is suitable to make a short, informative utterance (e.g. *"turn left here", "traffic jam ahead, turn left instead"*).

Finally, it is also important to consider the cognitive workload level of the user to determine when - and if - to resume, and also whether the topic that is to be resumed belongs to a domain where the system has information to give to the user, or a domain where the user gives information to the system. For example, if the user is using a navigation system and he or she is experiencing driving-induced workload when approaching e.g. a crossing, it might be a good idea to give additional navigation information even though the user has not explicitly asked for it. If the user however is using a telephone application it is probably better to let the user initiate the resumption. The DICO corpus shows that it is the passenger that is most careful not to interrupt or resume when the driver's workload is high, indicating that the system should let the user decide whether it is suitable to resume during high workload, while it is more accepted to let the system interrupt and resume when the workload is low.

When resuming to the interview domain the driver (i.e. the user) mostly uses declarative phrases, either as an answer to a question or as a redundant utterance to clarify what was last said before the interruption. Therefore the dialogue system should be able to store not only what has been agreed upon regarding the interrupted task, but also the last few utterances to make it possible to interpret the user utterance as a resumption.

It is common that the driver utterances are incomplete, perhaps due to the fact that the driver's primary task is the driving and therefore his or her mind is not always set on the dialogue task. Lindström (2008) showed that deletions are the most common disfluency during high cognitive load, which is supported by the results in this paper. The dialogue system should therefore be robust regarding ungrammatical utterances.

7 Future work

Next we intend to implement strategies for interruption and resumption in the DICO dialogue system. The strategies will then be evaluated through user tests where the participants will compare an application with these strategies with an application without them. Cognitive workload will be measured as well as driving ability (for example, by using a Lane Change Task (Mattes, 2003)). The participants will also be interviewed in order to find out which version of the system is more pleasant to use.

References

Robert Broström, Johan Engström, Anders Agnvall, and Gustav Markkula. 2006. Towards the next generation intelligent driver information system (idis): The volvo cars interaction manager concept. In *Proceedings of the 2006 ITS World Congress*.

Staffan Larsson and Jessica Villing. 2007. The dico project: A multimodal menu-based in-vehicle dialogue system. In H C Bunt and E C G Thijsse, editors, *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, page 4.

Anders Lindström, Jessica Villing, Staffan Larsson, Alexander Seward, Nina Åberg, and Cecilia Holtelius. 2008. The effect of cognitive load on disfluencies during in-vehicle spoken dialogue. In *Proceedings of Interspeech 2008*, page 4.

Stefan Mattes. 2003. The lane-change-task as a tool for driver distraction evaluation. In *Proceedings of IGfA*.

V L Neale, T A Dingus, S G Klauer, J Sudweeks, and M Goodman. 2005. An overview of the 100-car naturalistic study and findings. In *Proceedings of the 19th International Technical Conference on Enhanced Safety of Vehicles (ESV)*.

W van Winsum, M Martens, and L Herland. 1999. The effect of speech versus tactile driver support messages on workload, driver behaviour and user acceptance. tno-report tm-99-c043. Technical report, Soesterberg, Netherlands.

Jessica Villing, Cecilia Holtelius, Staffan Larsson, Anders Lindström, Alexander Seward, and Nina Åberg. 2008. Interruption, resumption and domain switching in in-vehicle dialogue. In *Proceedings of GoTAL, 6th International Conference on Natural Language Processing*, page 12.

Jessica Villing. 2009. In-vehicle dialogue management - towards distinguishing between different types of workload. In *Proceedings of SiMPE, Fourth Workshop on Speech in Mobile and Pervasive Environments*, pages 14–21.

Fan Yang and Peter A Heeman. 2009. Context restoration in multi-tasking dialogue. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 373–378, New York, NY, USA. ACM.

Learning to Follow Navigational Directions

Adam Vogel and **Dan Jurafsky**
Department of Computer Science
Stanford University
{acvogel,jurafsky}@stanford.edu

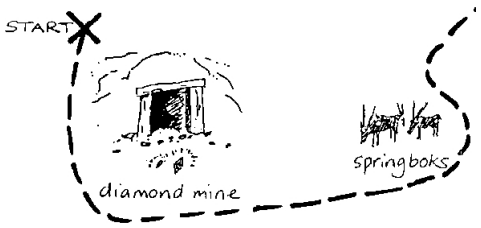
Abstract

We present a system that learns to follow navigational natural language directions. Where traditional models learn from linguistic annotation or word distributions, our approach is grounded in the world, learning by apprenticeship from routes through a map paired with English descriptions. Lacking an explicit alignment between the text and the reference path makes it difficult to determine what portions of the language describe which aspects of the route. We learn this correspondence with a reinforcement learning algorithm, using the deviation of the route we follow from the intended path as a reward signal. We demonstrate that our system successfully grounds the meaning of spatial terms like *above* and *south* into geometric properties of paths.

1 Introduction

Spatial language usage is a vital component for physically grounded language understanding systems. Spoken language interfaces to robotic assistants (Wei et al., 2009) and Geographic Information Systems (Wang et al., 2004) must cope with the inherent ambiguity in spatial descriptions.

The semantics of imperative and spatial language is heavily dependent on the physical setting it is situated in, motivating automated learning approaches to acquiring meaning. Traditional accounts of learning typically rely on linguistic annotation (Zettlemoyer and Collins, 2009) or word distributions (Curran, 2003). In contrast, we present an apprenticeship learning system which learns to imitate human instruction following, without linguistic annotation. Solved using a reinforcement learning algorithm, our system acquires the meaning of spatial words through grounded interaction with the world. This draws on the intuition that children learn to use spatial language through a mixture of observing adult language usage and situated interaction in the world, usually without explicit definitions (Tanz, 1980).



1. go vertically down until you're underneath eh diamond mine
2. then eh go right until you're
3. you're between springbok and highest viewpoint

Figure 1: A path appears on the instruction giver's map, who describes it to the instruction follower.

Our system learns to follow navigational directions in a route following task. We evaluate our approach on the HCRC Map Task corpus (Anderson et al., 1991), a collection of spoken dialogs describing paths to take through a map. In this setting, two participants, the *instruction giver* and *instruction follower*, each have a map composed of named landmarks. Furthermore, the *instruction giver* has a route drawn on her map, and it is her task to describe the path to the *instruction follower*, who cannot see the reference path. Our system learns to interpret these navigational directions, without access to explicit linguistic annotation.

We frame direction following as an apprenticeship learning problem and solve it with a reinforcement learning algorithm, extending previous work on interpreting instructions by Branavan et al. (2009). Our task is to learn a policy, or mapping

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 806–814,
Uppsala, Sweden, 11-16 July 2010.

from world state to action, which most closely follows the reference route. Our state space combines world and linguistic features, representing both our current position on the map and the communicative content of the utterances we are interpreting. During training we have access to the reference path, which allows us to measure the utility, or reward, for each step of interpretation. Using this reward signal as a form of supervision, we learn a policy to maximize the expected reward on unseen examples.

2 Related Work

Levit and Roy (2007) developed a spatial semantics for the Map Task corpus. They represent instructions as Navigational Information Units, which decompose the meaning of an instruction into orthogonal constituents such as the reference object, the type of movement, and quantitative aspect. For example, they represent the meaning of "move two inches toward the house" as a reference object (the house), a path descriptor (towards), and a quantitative aspect (two inches). These representations are then combined to form a path through the map. However, they do not learn these representations from text, leaving natural language processing as an open problem. The semantics in our paper is simpler, eschewing quantitative aspects and path descriptors, and instead focusing on reference objects and frames of reference. This simplifies the learning task, without sacrificing the core of their representation.

Learning to follow instructions by interacting with the world was recently introduced by Branavan et al. (2009), who developed a system which learns to follow Windows Help guides. Our reinforcement learning formulation follows closely from their work. Their approach can incorporate expert supervision into the reward function in a similar manner to this paper, but is also able to learn effectively from environment feedback alone. The Map Task corpus is free form conversational English, whereas the Windows instructions are written by a professional. In the Map Task corpus we only observe expert route following behavior, but are not told how portions of the text correspond to parts of the path, leading to a difficult learning problem.

The semantics of spatial language has been studied for some time in the linguistics literature. Talmy (1983) classifies the way spatial meaning is

Figure 2: The instruction giver and instruction follower face each other, and cannot see each others maps.

encoded syntactically, and Fillmore (1997) studies spatial terms as a subset of deictic language, which depends heavily on non-linguistic context. Levinson (2003) conducted a cross-linguistic semantic typology of spatial systems. Levinson categorizes the frames of reference, or spatial coordinate systems[1], into

1. *Egocentric*: Speaker/hearer centered frame of reference. Ex: "the ball to your left".

2. *Allocentric*: Speaker independent. Ex: "the road to the north of the house"

Levinson further classifies allocentric frames of reference into absolute, which includes the cardinal directions, and intrinsic, which refers to a featured side of an object, such as "the front of the car". Our spatial feature representation follows this egocentric/allocentric distinction. The intrinsic frame of reference occurs rarely in the Map Task corpus and is ignored, as speakers tend not to mention features of the landmarks beyond their names.

Regier (1996) studied the learning of spatial language from static 2-D diagrams, learning to distinguish between terms with a connectionist model. He focused on the meaning of individual terms, pairing a diagram with a given word. In contrast, we learn from whole texts paired with a

[1]Not all languages exhibit all frames of reference. Terms for 'up' and 'down' are exhibited in most all languages, while 'left' and 'right' are absent in some. Gravity breaks the symmetry between 'up' and 'down' but no such physical distinction exists for 'left' and 'right', which contributes to the difficulty children have learning them.

path, which requires learning the correspondence between text and world. We use similar geometric features as Regier, capturing the allocentric frame of reference.

Spatial semantics have also been explored in physically grounded systems. Kuipers (2000) developed the Spatial Semantic Hierarchy, a knowledge representation formalism for representing different levels of granularity in spatial knowledge. It combines sensory, metrical, and topological information in a single framework. Kuipers et al. demonstrate its effectiveness on a physical robot, but did not address the learning problem.

More generally, apprenticeship learning is well studied in the reinforcement learning literature, where the goal is to mimic the behavior of an expert in some decision making domain. Notable examples include (Abbeel and Ng, 2004), who train a helicopter controller from pilot demonstration.

3 The Map Task Corpus

The HCRC Map Task Corpus (Anderson et al., 1991) is a set of dialogs between an *instruction giver* and an *instruction follower*. Each participant has a map with small named landmarks. Additionally, the instruction giver has a path drawn on her map, and must communicate this path to the instruction follower in natural language. Figure 1 shows a portion of the instruction giver's map and a sample of the instruction giver language which describes part of the path.

The Map Task Corpus consists of 128 dialogs, together with 16 different maps. The speech has been transcribed and segmented into utterances, based on the length of pauses. We restrict our attention to just the utterances of the instruction giver, ignoring the instruction follower. This is to reduce redundancy and noise in the data - the instruction follower rarely introduces new information, instead asking for clarification or giving confirmation. The landmarks on the instruction follower map sometimes differ in location from the instruction giver's. We ignore this caveat, giving the system access to the instruction giver's landmarks, without the reference path.

Our task is to build an automated *instruction follower*. Whereas the original participants could speak freely, our system does not have the ability to query the instruction giver and must instead rely only on the previously recorded dialogs.

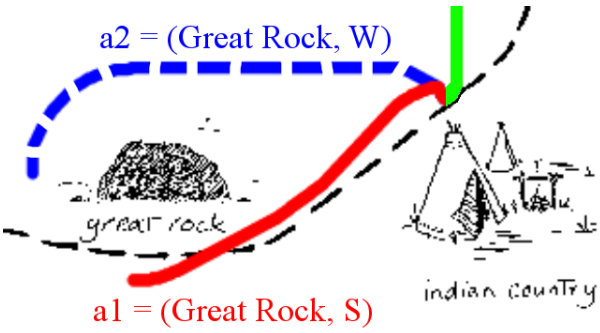


Figure 3: Sample state transition. Both actions get credit for visiting the great rock after the indian country. Action a1 also gets credit for passing the great rock on the correct side.

4 Reinforcement Learning Formulation

We frame the direction following task as a sequential decision making problem. We interpret utterances in order, where our interpretation is expressed by moving on the map. Our goal is to construct a series of moves in the map which most closely matches the expert path.

We define intermediate steps in our interpretation as *states* in a set S, and interpretive steps as *actions* drawn from a set A. To measure the fidelity of our path with respect to the expert, we define a *reward* function $R : S \times A \rightarrow \mathbb{R}^+$ which measures the utility of choosing a particular action in a particular state. Executing action a in state s carries us to a new state s', and we denote this transition function by $s' = T(s, a)$. All transitions are deterministic in this paper.[2]

For training we are given a set of dialogs D. Each dialog $d \in D$ is segmented into utterances $(u_1, \ldots, u_m)$ and is paired with a map, which is composed of a set of named landmarks $(l_1, \ldots, l_n)$.

4.1 State

The states of our decision making problem combine both our position in the dialog d and the path we have taken so far on the map. A state $s \in S$ is composed of $s = (u_i, l, c)$, where l is the named landmark we are located next to and c is a cardinal direction drawn from $\{\text{North}, \text{South}, \text{East}, \text{West}\}$ which determines which side of l we are on. Lastly, u_i is the utterance in d we are currently interpreting.

[2]Our learning algorithm is not dependent on a deterministic transition function and can be applied to domains with stochastic transitions, such as robot locomotion.

4.2 Action

An action $a \in A$ is composed of a named landmark l, the *target* of the action, together with a cardinal direction c which determines which side to pass l on. Additionally, a can be the null action, with $l = l'$ and $c = c'$. In this case, we interpret an utterance without moving on the map. A target l together with a cardinal direction c determine a point on the map, which is a fixed distance from l in the direction of c.

We make the assumption that at most one instruction occurs in a given utterance. This does not always hold true - the instruction giver sometimes chains commands together in a single utterance.

4.3 Transition

Executing action $a = (l', c')$ in state $s = (u_i, l, c)$ leads us to a new state $s' = T(s, a)$. This transition moves us to the next utterance to interpret, and moves our location to the target of the action. If a is the null action, $s = (u_{i+1}, l, c)$, otherwise $s' = (u_{i+1}, l', c')$. Figure 3 displays the state transitions two different actions.

To form a path through the map, we connect these state waypoints with a path planner[3] based on A^*, where the landmarks are obstacles. In a physical system, this would be replaced with a robot motion planner.

4.4 Reward

We define a reward function $R(s, a)$ which measures the utility of executing action a in state s. We wish to construct a route which follows the expert path as closely as possible. We consider a proposed route P close to the expert path P_e if P visits landmarks in the same order as P_e, and also passes them on the correct side.

For a given transition $s = (u_i, l, c)$, $a = (l', c')$, we have a binary feature indicating if the expert path moves from l to l'. In Figure 3, both a_1 and a_2 visit the next landmark in the correct order.

To measure if an action is to the correct side of a landmark, we have another binary feature indicating if P_e passes l' on side c. In Figure 3, only a_1 passes l' on the correct side.

In addition, we have a feature which counts the number of words in u_i which also occur in the name of l'. This encourages us to choose policies which interpret language relevant to a given landmark.

Our reward function is a linear combination of these features.

4.5 Policy

We formally define an interpretive strategy as a *policy* $\pi : S \rightarrow A$, a mapping from states to actions. Our goal is to find a policy π which maximizes the expected reward $\mathbb{E}_\pi[R(s, \pi(s))]$. The expected reward of following policy π from state s is referred to as the *value* of s, expressed as

$$V^\pi(s) = \mathbb{E}_\pi[R(s, \pi(s))] \quad (1)$$

When comparing the utilities of executing an action a in a state s, it is useful to define a function

$$\begin{aligned} Q^\pi(s, a) &= R(s, a) + V^\pi(T(s, a)) \\ &= R(s, a) + Q^\pi(T(s, a), \pi(s)) \end{aligned} \quad (2)$$

which measures the utility of executing a, and following the policy π for the remainder. A given Q function implicitly defines a policy π by

$$\pi(s) = \max_a Q(s, a). \quad (3)$$

Basic reinforcement learning methods treat states as atomic entities, in essence estimating V^π as a table. However, at test time we are following new directions for a map we haven't previously seen. Thus, we represent state/action pairs with a feature vector $\phi(s, a) \in \mathbb{R}^K$. We then represent the Q function as a linear combination of the features,

$$Q(s, a) = \theta^{\mathrm{T}} \phi(s, a) \quad (4)$$

and learn weights θ which most closely approximate the true expected reward.

4.6 Features

Our features $\phi(s, a)$ are a mixture of world and linguistic information. The linguistic information in our feature representation includes the instruction giver utterance and the names of landmarks on the map. Additionally, we furnish our algorithm with a list of English spatial terms, shown in Table 1. Our feature set includes approximately 200 features. Learning exactly which words influence decision making is difficult; reinforcement learning algorithms have problems with the large, sparse feature vectors common in natural language processing.

For a given state $s = (u, l, c)$ and action $a = (l', c')$, our feature vector $\phi(s, a)$ is composed of the following:

[3] We used the Java Path Planning Library, available at `http://www.cs.cmu.edu/~ggordon/PathPlan/`.

above, below, under, underneath, over, bottom, top, up, down, left, right, north, south, east, west, on

Table 1: The list of given spatial terms.

- **Coherence**: The number of words $w \in u$ that occur in the name of l'
- **Landmark Locality**: Binary feature indicating if l' is the closest landmark to l
- **Direction Locality**: Binary feature indicating if cardinal direction c' is the side of l' closest to (l, c)
- **Null Action**: Binary feature indicating if $l' =$ NULL
- **Allocentric Spatial**: Binary feature which conjoins the side c we pass the landmark on with each spatial term $w \in u$. This allows us to capture that the word *above* tends to indicate passing to the north of the landmark.
- **Egocentric Spatial**: Binary feature which conjoins the cardinal direction we move in with each spatial term $w \in u$. For instance, if (l, c) is above (l', c'), the direction from our current position is south. We conjoin this direction with each spatial term, giving binary features such as "the word *down* appears in the utterance and we move to the south".

5 Approximate Dynamic Programming

Given this feature representation, our problem is to find a parameter vector $\theta \in \mathbb{R}^K$ for which $Q(s, a) = \theta^{\mathrm{T}}\phi(s, a)$ most closely approximates $\mathbb{E}[R(s, a)]$. To learn these weights θ we use SARSA (Sutton and Barto, 1998), an online learning algorithm similar to Q-learning (Watkins and Dayan, 1992).

Algorithm 1 details the learning algorithm, which we follow here. We iterate over training documents $d \in D$. In a given state s_t, we act according to a probabilistic policy defined in terms of the Q function. After every transition we update θ, which changes how we act in subsequent steps.

Exploration is a key issue in any RL algorithm. If we act greedily with respect to our current Q function, we might never visit states which are actually higher in value. We utilize Boltzmann exploration, for which

$$\Pr(a_t|s_t;\theta) = \frac{\exp(\frac{1}{\tau}\theta^{\mathrm{T}}\phi(s_t, a_t))}{\sum_{a'}\exp(\frac{1}{\tau}\theta^{\mathrm{T}}\phi(s_t, a'))} \quad (5)$$

The parameter τ is referred to as the *temperature*, with a higher temperature causing more exploration, and a lower temperature causing more exploitation. In our experiments $\tau = 2$.

Acting with this exploration policy, we iterate through the training dialogs, updating our feature weights θ as we go. The update step looks at two successive state transitions. Suppose we are in state s_t, execute action a_t, receive reward $r_t = R(s_t, a_t)$, transition to state s_{t+1}, and there choose action a_{t+1}. The variables of interest are $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$, which motivates the name SARSA.

Our current estimate of the Q function is $Q(s, a) = \theta^{\mathrm{T}}\phi(s, a)$. By the Bellman equation, for the true Q function

$$Q(s_t, a_t) = R(s_t, a_t) + \max_{a'} Q(s_{t+1}, a') \quad (6)$$

After each action, we want to move θ to minimize the *temporal difference*,

$$R(s_t, a_t) + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \quad (7)$$

Input: Dialog set D
Reward function R
Feature function ϕ
Transition function T
Learning rate α_t
Output: Feature weights θ

Initialize θ to small random values
until θ *converges* **do**
  foreach *Dialog* $d \in D$ **do**
    Initialize $s_0 = (l_1, u_1, \emptyset)$,
    $a_0 \sim \Pr(a_0|s_0;\theta)$
    for $t = 0$*;* s_t *non-terminal;* $t{+}{+}$ **do**
      Act: $s_{t+1} = T(s_t, a_t)$
      Decide: $a_{t+1} \sim \Pr(a_{t+1}|s_{t+1};\theta)$
      Update:
      $\Delta \leftarrow R(s_t, a_t) + \theta^{\mathrm{T}}\phi(s_{t+1}, a_{t+1})$
        $- \theta^{\mathrm{T}}\phi(s_t, a_t)$
      $\theta \leftarrow \theta + \alpha_t\phi(s_t, a_t)\Delta$
    end
  end
end
return θ

Algorithm 1: The SARSA learning algorithm.

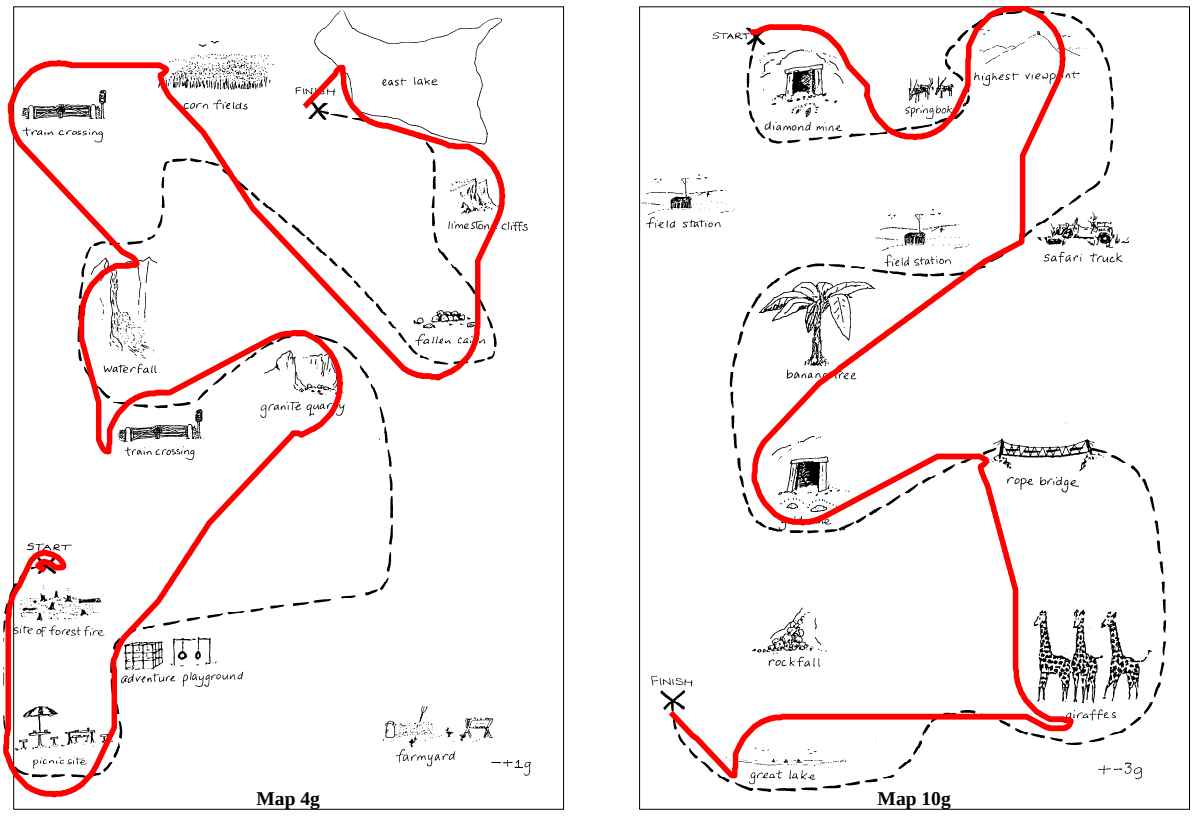


Figure 4: Sample output from the SARSA policy. The dashed black line is the reference path and the solid red line is the path the system follows.

For each feature $\phi_i(s_t, a_t)$, we change θ_i proportional to this temporal difference, tempered by a learning rate α_t. We update θ according to

$$\theta = \theta + \alpha_t \phi(s_t, a_t)(R(s_t, a_t) + \theta^{\mathrm{T}} \phi(s_{t+1}, a_{t+1}) - \theta^{\mathrm{T}} \phi(s_t, a_t)) \quad (8)$$

Here α_t is the learning rate, which decays over time[4]. In our case, $\alpha_t = \frac{10}{10+t}$, which was tuned on the training set. We determine convergence of the algorithm by examining the magnitude of updates to θ. We stop the algorithm when

$$||\theta_{t+1} - \theta_t||_\infty < \epsilon \quad (9)$$

6 Experimental Design

We evaluate our system on the Map Task corpus, splitting the corpus into 96 training dialogs and 32 test dialogs. The whole corpus consists of approximately 105,000 word tokens. The maps seen at test time do not occur in the training set, but some of the human participants are present in both.

[4]To guarantee convergence, we require $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$. Intuitively, the sum diverging guarantees we can still learn arbitrarily far into the future, and the sum of squares converging guarantees that our updates will converge at some point.

6.1 Evaluation

We evaluate how closely the path P generated by our system follows the expert path P_e. We measure this with respect to two metrics: the order in which we visit landmarks and the side we pass them on.

To determine the order P_e visits landmarks we compute the minimum distance from P_e to each landmark, and threshold it at a fixed value.

To score path P, we compare the order it visits landmarks to the expert path. A transition $l \rightarrow l'$ which occurs in P counts as correct if the same transition occurs in P_e. Let $|P|$ be the number of landmark transitions in a path P, and N the number of correct transitions in P. We define the *order precision* as $N/|P|$, and the *order recall* as $N/|P_e|$.

We also evaluate how well we are at passing landmarks on the correct side. We calculate the distance of P_e to each side of the landmark, considering the path to visit a side of the landmark if the distance is below a threshold. This means that a path might be considered to visit multiple sides of a landmark, although in practice it is usu-

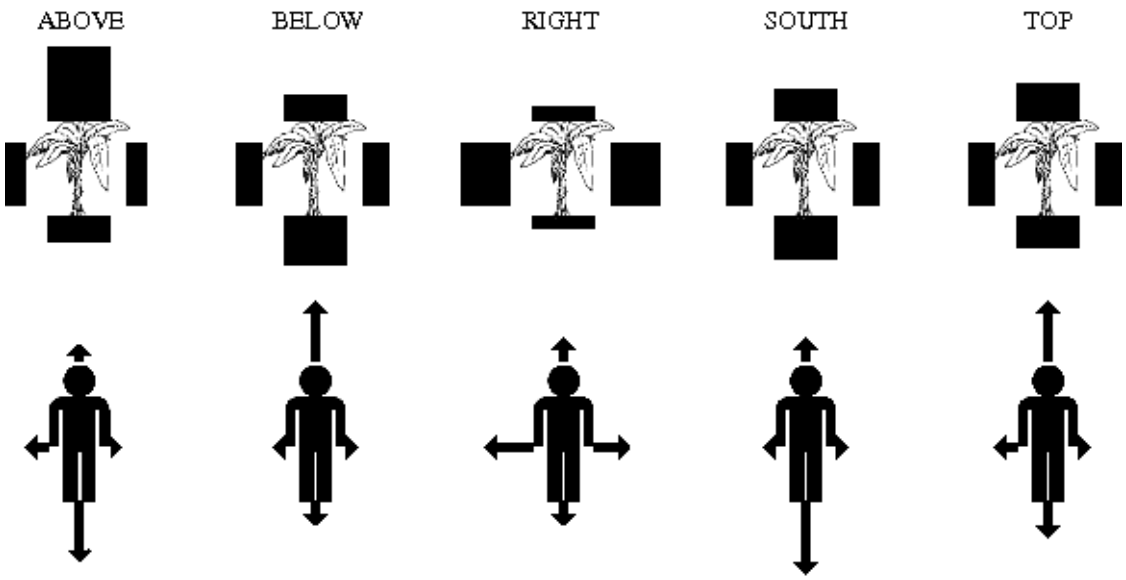


Figure 5: This figure shows the relative weights of spatial features organized by spatial word. The top row shows the weights of allocentric (landmark-centered) features. For example, the top left figure shows that when the word *above* occurs, our policy prefers to go to the north of the target landmark. The bottom row shows the weights of egocentric (absolute) spatial features. The bottom left figure shows that given the word *above*, our policy prefers to move in a southerly cardinal direction.

ally one. If C is the number of landmarks we pass on the correct side, define the *side precision* as $C/|P|$, and the *side recall* as $C/|P_e|$.

6.2 Comparison Systems

The baseline policy simply visits the closest landmark at each step, taking the side of the landmark which is closest. It pays no attention to the direction language.

We also compare against the policy gradient learning algorithm of Branavan et al. (2009). They parametrize a probabilistic policy $\Pr(s|a;\theta)$ as a log-linear model, in a similar fashion to our exploration policy. During training, the learning algorithm adjusts the weights θ according to the gradient of the value function defined by this distribution.

Reinforcement learning algorithms can be classified into *value* based and *policy* based. Value methods estimate a value function V for each state, then act greedily with respect to it. Policy learning algorithms directly search through the space of policies. SARSA is a value based method, and the policy gradient algorithm is policy based.

	Visit Order			Side		
	P	R	F_1	P	R	F_1
Baseline	28.4	37.2	32.2	46.1	60.3	52.2
PG	31.1	43.9	36.4	49.5	**69.9**	57.9
SARSA	**45.7**	**51.0**	**48.2**	**58.0**	64.7	**61.2**

Table 2: Experimental results. Visit order shows how well we follow the order in which the answer path visits landmarks. 'Side' shows how successfully we pass on the correct side of landmarks.

7 Results

Table 2 details the quantitative performance of the different algorithms. Both SARSA and the policy gradient method outperform the baseline, but still fall significantly short of expert performance. The baseline policy performs surprisingly well, especially at selecting the correct side to visit a landmark.

The disparity between learning approaches and gold standard performance can be attributed to several factors. The language in this corpus is conversational, frequently ungrammatical, and contains troublesome aspects of dialog such as conversational repairs and repetition. Secondly, our action and feature space are relatively primitive, and don't capture the full range of spatial expression. Path descriptors, such as the difference between *around* and *past* are absent, and our feature

representation is relatively simple.

The SARSA learning algorithm accrues more reward than the policy gradient algorithm. Like most gradient based optimization methods, policy gradient algorithms oftentimes get stuck in local maxima, and are sensitive to the initial conditions. Furthermore, as the size of the feature vector K increases, the space becomes even more difficult to search. There are no guarantees that SARSA has reached the best policy under our feature space, and this is difficult to determine empirically. Thus, some accuracy might be gained by considering different RL algorithms.

8 Discussion

Examining the feature weights θ sheds some light on our performance. Figure 5 shows the relative strength of weights for several spatial terms. Recall that the two main classes of spatial features in ϕ are egocentric (what direction we move in) and allocentric (on which side we pass a landmark), combined with each spatial word.

Allocentric terms such as *above* and *below* tend to be interpreted as going to the north and south of landmarks, respectively. Interestingly, our system tends to move in the opposite cardinal direction, i.e. the agent moves south in the egocentric frame of reference. This suggests that people use *above* when we are already above a landmark. *South* slightly favors passing on the south side of landmarks, and has a heavy tendency to move in a southerly direction. This suggests that south is used more frequently in an egocentric reference frame.

Our system has difficulty learning the meaning of *right*. *Right* is often used as a conversational filler, and also for dialog alignment, such as

> "right okay right go vertically up then between the springboks and the highest viewpoint."

Furthermore, *right* can be used in both an egocentric or allocentric reference frame. Compare

> "go to the uh right of the mine"

which utilizes an allocentric frame, with

> "right then go eh uh to your right horizontally"

which uses an egocentric frame of reference. It is difficult to distinguish between these meanings without syntactic features.

9 Conclusion

We presented a reinforcement learning system which learns to interpret natural language directions. Critically, our approach uses no semantic annotation, instead learning directly from human demonstration. It successfully acquires a subset of spatial semantics, using reinforcement learning to derive the correspondence between instruction language and features of paths. While our results are still preliminary, we believe our model represents a significant advance in learning natural language meaning, drawing its supervision from human demonstration rather than word distributions or hand-labeled semantic tags. Framing language acquisition as apprenticeship learning is a fruitful research direction which has the potential to connect the symbolic, linguistic domain to the non-symbolic, sensory aspects of cognition.

Acknowledgments

This research was partially supported by the National Science Foundation via a Graduate Research Fellowship to the first author and award IIS-0811974 to the second author and by the Air Force Research Laboratory (AFRL), under prime contract no. FA8750-09-C-0181. Thanks to Michael Levit and Deb Roy for providing digital representations of the maps and a subset of the corpus annotated with their spatial representation.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press.

A. Anderson, M. Bader, E. Bard, E. Boyle, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. Mcallister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert. 1991. The HCRC map task corpus. *Language and Speech, 34*, pages 351–366.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *ACL-IJCNLP '09*.

James Richard Curran. 2003. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Charles Fillmore. 1997. *Lectures on Deixis*. Stanford: CSLI Publications.

Benjamin Kuipers. 2000. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233.

Stephen Levinson. 2003. *Space In Language And Cognition: Explorations In Cognitive Diversity*. Cambridge University Press.

Michael Levit and Deb Roy. 2007. Interpretation of spatial language in a map navigation task. In *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37(3)*, pages 667–679.

Terry Regier. 1996. *The Human Semantic Potential: Spatial Language and Constrained Connectionism*. The MIT Press.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Leonard Talmy. 1983. How language structures space. In *Spatial Orientation: Theory, Research, and Application*.

Christine Tanz. 1980. *Studies in the acquisition of deictic terms*. Cambridge University Press.

Hongmei Wang, Alan M. Maceachren, and Guoray Cai. 2004. Design of human-GIS dialogue for communication of vague spatial concepts. In *GIScience*.

C. J. C. H. Watkins and P. Dayan. 1992. Q-learning. *Machine Learning*, pages 8:279–292.

Yuan Wei, Emma Brunskill, Thomas Kollar, and Nicholas Roy. 2009. Where to go: interpreting natural directions using global inference. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 3761–3767, Piscataway, NJ, USA. IEEE Press.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *ACL-IJCNLP '09*, pages 976–984.

A Hybrid Hierarchical Model for Multi-Document Summarization

Asli Celikyilmaz
Computer Science Department
University of California, Berkeley
asli@eecs.berkeley.edu

Dilek Hakkani-Tur
International Computer Science Institute
Berkeley, CA
dilek@icsi.berkeley.edu

Abstract

Scoring sentences in documents given abstract summaries created by humans is important in extractive multi-document summarization. In this paper, we formulate extractive summarization as a two step learning problem building a generative model for pattern discovery and a regression model for inference. We calculate scores for sentences in document clusters based on their latent characteristics using a hierarchical topic model. Then, using these scores, we train a regression model based on the lexical and structural characteristics of the sentences, and use the model to score sentences of new documents to form a summary. Our system advances current state-of-the-art improving ROUGE scores by ∼7%. Generated summaries are less redundant and more coherent based upon manual quality evaluations.

1 Introduction

Extractive approach to multi-document summarization (MDS) produces a summary by selecting sentences from original documents. Document Understanding Conferences (DUC), now TAC, fosters the effort on building MDS systems, which take document clusters (documents on a same topic) and description of the desired summary focus as input and output a word length limited summary. Human summaries are provided for training summarization models and measuring the performance of machine generated summaries.

Extractive summarization methods can be classified into two groups: supervised methods that rely on provided document-summary pairs, and unsupervised methods based upon properties derived from document clusters. Supervised methods treat the summarization task as a classification/regression problem, e.g., (Shen et al., 2007; Yeh et al., 2005). Each candidate sentence is classified as summary or non-summary based on the features that they pose and those with highest scores are selected. Unsupervised methods aim to score sentences based on semantic groupings extracted from documents, e.g., (Daumé III and Marcu, 2006; Titov and McDonald, 2008; Tang et al., 2009; Haghighi and Vanderwende, 2009; Radev et al., 2004; Branavan et al., 2009), etc. Such models can yield comparable or better performance on DUC and other evaluations, since representing documents as topic distributions rather than bags of words diminishes the effect of lexical variability. To the best of our knowledge, there is no previous research which utilizes the best features of both approaches for MDS as presented in this paper.

In this paper, we present a novel approach that formulates MDS as a prediction problem based on a two-step hybrid model: a generative model for hierarchical topic discovery and a regression model for inference. We investigate if a hierarchical model can be adopted to discover salient characteristics of sentences organized into hierarchies utilizing human generated summary text.

We present a probabilistic topic model on sentence level building on hierarchical Latent Dirichlet Allocation (hLDA) (Blei et al., 2003a), which is a generalization of LDA (Blei et al., 2003b). We construct a hybrid learning algorithm by extracting salient features to characterize summary sentences, and implement a regression model for inference (Fig.3). Contributions of this work are:
− construction of hierarchical probabilistic model designed to discover the topic structures of all sentences. Our focus is on identifying similarities of candidate sentences to summary sentences using a novel tree based sentence scoring algorithm, concerning topic distributions at different levels of the discovered hierarchy as described in § 3 and § 4,
− representation of sentences by meta-features to

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 815–824,
Uppsala, Sweden, 11-16 July 2010.

characterize their candidacy for inclusion in summary text. Our aim is to find features that can best represent summary sentences as described in § 5,
– implementation of a feasible inference method based on a regression model to enable scoring of sentences in test document clusters without retraining, (which has not been investigated in generative summarization models) described in § 5.2.

We show in § 6 that our hybrid summarizer achieves comparable (if not better) ROUGE score on the challenging task of extracting the summaries of multiple newswire documents. The human evaluations confirm that our hybrid model can produce coherent and non-redundant summaries.

2 Background and Motivation

There are many studies on the principles governing multi-document summarization to produce coherent and semantically relevant summaries. Previous work (Nenkova and Vanderwende, 2005; Conroy et al., 2006), focused on the fact that frequency of words plays an important factor. While, earlier work on summarization depend on a word score function, which is used to measure sentence rank scores based on (semi-)supervised learning methods, recent trend of purely data-driven methods, (Barzilay and Lee, 2004; Daumé III and Marcu, 2006; Tang et al., 2009; Haghighi and Vanderwende, 2009), have shown remarkable improvements. Our work builds on both methods by constructing a hybrid approach to summarization.

Our objective is to discover from document clusters, the latent topics that are organized into hierarchies following (Haghighi and Vanderwende, 2009). A hierarchical model is particularly appealing to summarization than a "flat" model, e.g. LDA (Blei et al., 2003b), in that one can discover "abstract" and "specific" topics. For instance, discovering that "baseball" and "football" are both contained in an abstract class "sports" can help to identify summary sentences. It follows that summary topics are commonly shared by many documents, while specific topics are more likely to be mentioned in rather a small subset of documents.

Feature based learning approaches to summarization methods discover salient features by measuring similarity between candidate sentences and summary sentences (Nenkova and Vanderwende, 2005; Conroy et al., 2006). While such methods are effective in extractive summarization, the fact that some of these methods are based on greedy algorithms can limit the application areas. Moreover, using information on the hidden semantic structure of document clusters would improve the performance of these methods.

Recent studies focused on the discovery of latent topics of document sets in extracting summaries. In these models, the challenges of inferring topics of test documents are not addressed in detail. One of the challenges of using a previously trained topic model is that the new document might have a totally new vocabulary or may include many other specific topics, which may or may not exist in the trained model. A common method is to re-build a topic model for new sets of documents (Haghighi and Vanderwende, 2009), which has proven to produce coherent summaries. An alternative yet feasible solution, presented in this work, is building a model that can summarize new document clusters using characteristics of topic distributions of training documents. Our approach differs from the early work, in that, we combine a generative hierarchical model and regression model to score sentences in new documents, eliminating the need for building a generative model for new document clusters.

3 Summary-Focused Hierarchical Model

Our MDS system, hybrid hierarchical summarizer, **HybHSum**, is based on an hybrid learning approach to extract sentences for generating summary. We discover hidden topic distributions of sentences in a given document cluster along with provided summary sentences based on hLDA described in (Blei et al., 2003a)[1]. We build a summary-focused hierarchical probabilistic topic model, sumHLDA, for each document cluster at sentence level, because it enables capturing expected topic distributions in given sentences directly from the model. Besides, document clusters contain a relatively small number of documents, which may limit the variability of topics if they are evaluated on the document level. As described in § 4, we present a new method for scoring candidate sentences from this hierarchical structure.

Let a given document cluster D be represented with sentences O=$\{o_m\}_{m=1}^{|O|}$ and its corresponding human summary be represented with sentences S=$\{s_n\}_{n=1}^{|S|}$. All sentences are comprised of words $V = \{w_1, w_2, ..w_{|V|}\}$ in $\{O \cup S\}$.

[1]Please refer to (Blei et al., 2003b) and (Blei et al., 2003a) for details and demonstrations of topic models.

Summary hLDA (sumHLDA): The hLDA represents distribution of topics in sentences by organizing topics into a tree of a fixed depth L (Fig.1.a). Each candidate sentence o_m is assigned to a path c_{o_m} in the tree and each word w_i in a given sentence is assigned to a hidden topic z_{o_m} at a level l of c_{o_m}. Each node is associated with a topic distribution over words. The sampler method alternates between choosing a new path for each sentence through the tree and assigning each word in each sentence to a topic along that path. The structure of tree is learnt along with the topics using a nested Chinese restaurant process (nCRP) (Blei et al., 2003a), which is used as a prior.

The nCRP is a stochastic process, which assigns probability distributions to infinitely branching and infinitely deep trees. In our model, nCRP specifies a distribution of words into paths in an L-level tree. The assignments of sentences to paths are sampled sequentially: The first sentence takes the initial L-level path, starting with a single branch tree. Later, mth subsequent sentence is assigned to a path drawn from the distribution:

$$\begin{aligned} p(path_{old}, c|m, m_c) &= \frac{m_c}{\gamma+m-1} \\ p(path_{new}, c|m, m_c) &= \frac{\gamma}{\gamma+m-1} \end{aligned} \quad (1)$$

$path_{old}$ and $path_{new}$ represent an existing and novel (branch) path consecutively, m_c is the number of previous sentences assigned to path c, m is the total number of sentences seen so far, and γ is a hyper-parameter which controls the probability of creating new paths. Based on this probability each node can branch out a different number of child nodes proportional to γ. Small values of γ suppress the number of branches.

Summary sentences generally comprise abstract concepts of the content. With sumHLDA we want to capture these abstract concepts in candidate sentences. The idea is to represent each path shared by similar candidate sentences with representative summary sentence(s). We let summary sentences share existing paths generated by similar candidate sentences instead of sampling new paths and influence the tree structure by introducing two separate hyper-parameters for nCRP prior:

- if a summary sentence is sampled, use $\gamma = \gamma_s$,
- if a candidate sentence is sampled, use $\gamma = \gamma_o$.

At each node, we let summary sentences sample a path by choosing only from the existing children of that node with a probability proportional to the number of other sentences assigned to that child. This can be achieved by using a small value for γ_s ($0 < \gamma_s \lll 1$). We only let candidate sentences to have an option of creating a new child node with a probability proportional to γ_o. By choosing $\gamma_s \lll \gamma_o$ we suppress the generation of new branches for summary sentences and modify the γ of nCRP prior in Eq.(1) using γ_s and γ_o hyper-parameters for different sentence types. In the experiments, we discuss the effects of this modification on the hierarchical topic tree.

The following is the generative process for sumHLDA used in our `HybHSum` :

(1) For each topic $k \in T$, sample a distribution $\beta_k \backsim$ Dirichlet(η).

(2) For each sentence $d \in \{O \cup S\}$,

(a) if $d \in O$, draw a path $c_d \backsim$ nCRP(γ_o), else if $d \in S$, draw a path $c_d \backsim$ nCRP(γ_s).

(b) Sample L-vector θ_d mixing weights from Dirichlet distribution $\theta_d \sim Dir(\alpha)$.

(c) For each word n, choose: (*i*) level $z_{d,n}|\theta_d$ and (*ii*) word $w_{d,n}|\{z_{d,n}, c_d, \beta\}$

Given sentence d, θ_d is a vector of topic proportions from L dimensional Dirichlet parameterized by α (distribution over levels in the tree.) The nth word of d is sampled by first choosing a level $z_{d,n} = l$ from the discrete distribution θ_d with probability $\theta_{d,l}$. Dirichlet parameter η and γ_o control the size of tree effecting the number of topics. (Small values of γ_s do not effect the tree.) Large values of η favor more topics (Blei et al., 2003a).

Model Learning: Gibbs sampling is a common method to fit the hLDA models. The aim is to obtain the following samples from the posterior of: (*i*) the latent tree T, (*ii*) the level assignment **z** for all words, (*iii*) the path assignments **c** for all sentences conditioned on the observed words **w**.

Given the assignment of words **w** to levels **z** and assignments of sentences to paths **c**, the expected posterior probability of a particular word w at a given topic **z**=l of a path **c**=c is proportional to the number of times w was generated by that topic:

$$p(w|\mathbf{z}, \mathbf{c}, \mathbf{w}, \eta) \propto n_{(\mathbf{z}=l, \mathbf{c}=c, \mathbf{w}=w)} + \eta \quad (2)$$

Similarly, posterior probability of a particular topic z in a given sentence d is proportional to number of times z was generated by that sentence:

$$p(z|\mathbf{z}, \mathbf{c}, \alpha) \propto n_{(\mathbf{c}=c_d, \mathbf{z}=l)} + \alpha \quad (3)$$

$n_{(.)}$ is the count of elements of an array satisfying the condition. Note from Eq.(3) that two sentences d_1 and d_2 on the same path **c** would have

different words, and hence different posterior topic probabilities. Posterior probabilities are normalized with total counts and their hyperparameters.

4 Tree-Based Sentence Scoring

The sumHLDA constructs a hierarchical tree structure of candidate sentences (per document cluster) by positioning summary sentences on the tree. Each sentence is represented by a path in the tree, and each path can be shared by many sentences. The assumption is that sentences sharing the same path should be more similar to each other because they share the same topics. Moreover, if a path includes a summary sentence, then candidate sentences on that path are more likely to be selected for summary text. In particular, the similarity of a candidate sentence o_m to a summary sentence s_n sharing the same path is a measure of strength, indicating how likely o_m is to be included in the generated summary (Algorithm 1):

Let c_{o_m} be the path for a given o_m. We find summary sentences that share the same path with o_m via: $M = \{s_n \in S | c_{s_n} = c_{o_m}\}$. The score of each sentence is calculated by similarity to the best matching summary sentence in M:

$$score(o_m) = \max_{s_n \in M} sim(o_m, s_n) \quad (4)$$

If *M*=∅, then *score(*o_m*)*=∅. The efficiency of our similarity measure in identifying the best matching summary sentence, is tied to how expressive the extracted topics of our sumHLDA models are. Given path c_{o_m}, we calculate the similarity of o_m to each s_n, *n*=1..$|M|$ by measuring similarities on:

⋆ **sparse unigram distributions** (sim_1) at each topic l on c_{o_m}: similarity between $p(\mathbf{w}_{o_m,l}|\mathbf{z}_{o_m} = l, c_{o_m}, v_l)$ and $p(\mathbf{w}_{s_n,l}|\mathbf{z}_{s_n} = l, c_{o_m}, v_l)$

⋆⋆ **distributions of topic proportions** (sim_2); similarity between $p(\mathbf{z}_{o_m}|c_{o_m})$ and $p(\mathbf{z}_{s_n}|c_{o_m})$.

− ***sim***$_1$***:*** We define two sparse (discrete) unigram distributions for candidate o_m and summary s_n at each node l on a vocabulary identified with words generated by the topic at that node, $v_l \subset V$. Given $\mathbf{w}_{o_m} = \{w_1, ..., w_{|o_m|}\}$, let $\mathbf{w}_{o_m,l} \subset \mathbf{w}_{o_m}$ be the set of words in o_m that are generated from topic $\mathbf{z}_{o_m}$ at level l on path c_{o_m}. The discrete unigram distribution $p_{o_m l} = p(\mathbf{w}_{o_m,l}|\mathbf{z}_{o_m} = l, c_{o_m}, v_l)$ represents the probability over all words v_l assigned to topic $\mathbf{z}_{o_m}$ at level l, by sampling only for words in $\mathbf{w}_{o_m,l}$. Similarly, $p_{s_n,l} = p(\mathbf{w}_{s_n,l}|\mathbf{z}_{s_n}, c_{o_m}, v_l)$ is the probability of words $\mathbf{w}_{s_n}$ in s_n of the same topic. The probability of each word in $p_{o_m,l}$ and $p_{s_n,l}$ are obtained using Eq. (2) and then normalized (see Fig.1.b).

Algorithm 1 Tree-Based Sentence Scoring

```
Given tree T from sumHLDA, candidate and summary
  sentences:  O = {o_1, ..., o_m} , S = {s_1, ..., s_n}
for sentences m ← 1, ..., |O| do
    - Find path c_{o_m} on tree T and summary sentences
    on path c_{o_m}: M = {s_n ∈ S | c_{s_n} = c_{o_m}}
    for summary sentences n ← 1, ..., |M| do
        - Find score(o_m)=max_{s_n} sim(o_m, s_n),
        where sim(o_m, s_n) = sim_1 * sim_2
        using Eq.(7) and Eq.(8)
    end for
end for
Obtain scores Y = {score(o_m)}_{m=1}^{|O|}
```

The similarity between $p_{o_m,l}$ and $p_{s_n,l}$ is obtained by first calculating the divergence with *information radius- IR* based on Kullback-Liebler(KL) divergence, *p*=$p_{o_m,l}$, *q*=$p_{s_n,l}$:

$$IR_{c_{o_m},l}(p_{o_m,l}, p_{s_n,l}) = KL\left(p||\tfrac{p+q}{2}\right) + KL\left(q||\tfrac{p+q}{2}\right) \quad (5)$$

where, $KL(p||q) = \sum_i p_i \log \frac{p_i}{q_i}$. Then the divergence is transformed into a similarity measure (Manning and Schuetze, 1999):

$$W_{c_{o_m},l}(p_{o_m,l}, p_{s_n,l}) = 10^{-IR_{c_{o_m},l}(p_{o_m,l}, p_{s_n,l})} \quad (6)$$

IR is a measure of total divergence from the average, representing how much information is lost when two distributions p and q are described in terms of average distributions. We opted for *IR* instead of the commonly used *KL* because with *IR* there is no problem with infinite values since $\frac{p_i+q_i}{2} \neq 0$ if either $p_i \neq 0$ or $q_i \neq 0$. Moreover, unlike *KL*, *IR* is symmetric, i.e., *KL(p,q)≠KL(q,p)*.

Finally sim_1 is obtained by average similarity of sentences using Eq.(6) at each level of c_{o_m} by:

$$sim_1(o_m, s_n) = \frac{1}{L} \sum_{l=1}^{L} W_{c_{o_m},l}(p_{o_m,l}, p_{s_n,l}) * l \quad (7)$$

The similarity between $p_{o_m,l}$ and $p_{s_n,l}$ at each level is weighted proportional to the level l because the similarity between sentences should be rewarded if there is a specific word overlap at child nodes.

−***sim***$_2$***:*** We introduce another measure based on sentence-topic mixing proportions to calculate the concept-based similarities between o_m and s_n. We calculate the topic proportions of o_m and s_n, represented by $p_{z_{o_m}} = p(\mathbf{z}_{o_m}|c_{o_m})$ and $p_{z_{s_n}} = p(\mathbf{z}_{s_n}|c_{o_m})$ via Eq.(3). The similarity between the distributions is then measured with transformed IR

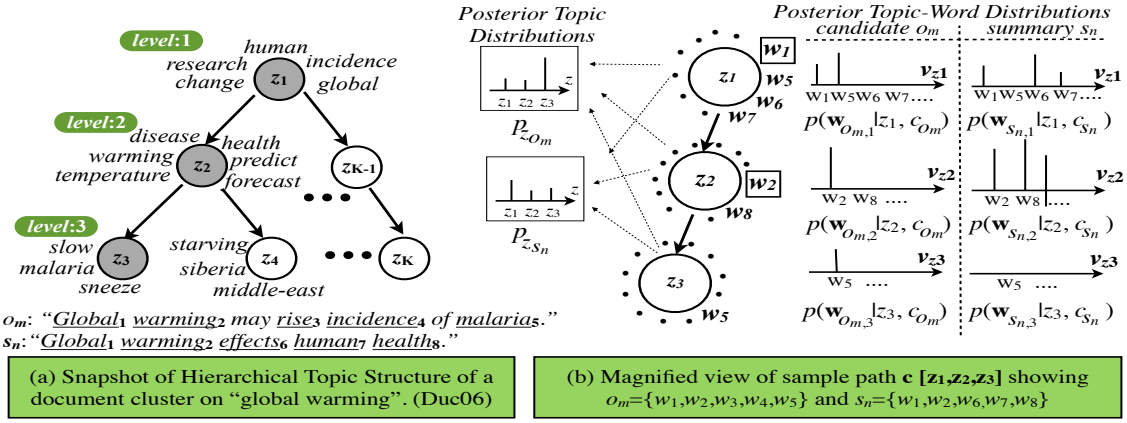


Figure 1: (a) A sample 3-level tree using sumHLDA. Each sentence is associated with a path c through the hierarchy, where each node $z_{l,c}$ is associated with a distribution over terms (Most probable terms are illustrated). (b) magnified view of a path (darker nodes) in (a). Distribution of words in given two sentences, a candidate (o_m) and a summary (s_n) using sub-vocabulary of words at each topic v_{z_l}. Discrete distributions on the left are topic mixtures for each sentence, $p_{z_{o_m}}$ and $p_{z_{s_n}}$.

as in Eq.(6) by:

$$sim_2(o_m, s_n) = 10^{-IR_{c_{o_m}}(p_{z_{o_m}}, p_{z_{s_n}})} \quad (8)$$

sim_1 provides information about the similarity between two sentences, o_m and s_n based on topic-word distributions. Similarly, sim_2 provides information on the similarity between the weights of the topics in each sentence. They jointly effect the sentence score and are combined in one measure:

$$sim(o_m, s_n) = sim_1(o_m, s_n) * sim_2(o_m, s_n) \quad (9)$$

The final score for a given o_m is calculated from Eq.(4). Fig.1.b depicts a sample path illustrating sparse unigram distributions of o_m and s_m at each level as well as their topic proportions, $p_{z_{o_m}}$, and $p_{z_{s_n}}$. In experiment 3, we discuss the effect of our tree-based scoring on summarization performance in comparison to a classical scoring method presented as our *baseline* model.

5 Regression Model

Each candidate sentence o_m, $m = 1..|O|$ is represented with a multi-dimensional vector of q features $\mathbf{f}_m = \{f_{m1}, ..., f_{mq}\}$. We build a regression model using sentence scores as output and selected salient features as input variables described below:

5.1 Feature Extraction

We compile our training dataset using sentences from different document clusters, which do not necessarily share vocabularies. Thus, we create n-gram *meta*-features to represent sentences instead of word n-gram frequencies:

(I) nGram Meta-Features (NMF): For each document cluster D, we identify most frequent (non-stop word) unigrams, i.e., $v_{freq} = \{w_i\}_{i=1}^{r} \subset V$, where r is a model parameter of number of most frequent unigram features. We measure observed unigram probabilities for each $w_i \in v_{freq}$ with $p_D(w_i) = n_D(w_i)/\sum_{j=1}^{|V|} n_D(w_j)$, where $n_D(w_i)$ is the number of times w_i appears in D and $|V|$ is the total number of unigrams. For any ith feature, the value is $f_{mi} = 0$, if given sentence does not contain w_i, otherwise $f_{mi} = p_D(w_i)$. These features can be extended for any n-grams. We similarly include bigram features in the experiments.

(II) Document Word Frequency Meta-Features (DMF): The characteristics of sentences at the document level can be important in summary generation. DMF identify whether a word in a given sentence is specific to the document in consideration or it is commonly used in the document cluster. This is important because summary sentences usually contain abstract terms rather than specific terms.

To characterize this feature, we re-use the r most frequent unigrams, i.e., $w_i \in v_{freq}$. Given sentence o_m, let d be the document that o_m belongs to, i.e., $o_m \in d$. We measure unigram probabilities for each w_i by $p(w_i \in o_m) = n_d(w_i \in o_m)/n_D(w_i)$, where $n_d(w_i \in o_m)$ is the number of times w_i appears in d and $n_D(w_i)$ is the number of times w_i appears in D. For any ith feature, the value is $f_{mi} = 0$, if given sentence does not contain w_i, otherwise $f_{mi} = p(w_i \in o_m)$. We also include bigram extensions of DMF features.

(III) Other Features (OF): Term frequency of sentences such as SUMBASIC are proven to be good predictors in sentence scoring (Nenkova and Vanderwende, 2005). We measure the average unigram probability of a sentence by: $p(o_m) = \sum_{w \in o_m} \frac{1}{|o_m|} P_D(w)$, where $P_D(w)$ is the observed unigram probability in the document collection D and $|o_m|$ is the total number of words in o_m. We use sentence bigram frequency, sentence rank in a document, and sentence size as additional features.

5.2 Predicting Scores for New Sentences

Due to the large feature space to explore, we chose to work with support vector regression (SVR) (Drucker et al., 1997) as the learning algorithm to predict sentence scores. Given training sentences $\{\mathbf{f}_m, y_m\}_{m=1}^{|O|}$, where $f_m = \{f_{m1}, ..., f_{mq}\}$ is a multi-dimensional vector of features and y_m=*score*$(o_m) \in \mathbb{R}$ are their scores obtained via Eq.(4), we train a regression model. In experiments we use non-linear Gaussian kernel for SVR. Once the SVR model is trained, we use it to predict the scores of n_{test} number of sentences in test (*unseen*) document clusters, $O_{test} = \{o_1, ...o_{|O_{test}|}\}$.

Our HybHSum captures the sentence characteristics with a regression model using sentences in different document clusters. At test time, this valuable information is used to score testing sentences.

Redundancy Elimination: To eliminate redundant sentences in the generated summary, we incrementally add onto the summary the highest ranked sentence o_m and check if o_m significantly repeats the information already included in the summary until the algorithm reaches word count limit. We use a word overlap measure between sentences normalized to sentence length. A o_m is discarded if its similarity to any of the previously selected sentences is greater than a threshold identified by a greedy search on the training dataset.

6 Experiments and Discussions

In this section we describe a number of experiments using our hybrid model on 100 document clusters each containing 25 news articles from DUC2005-2006 tasks. We evaluate the performance of HybHSum using 45 document clusters each containing 25 news articles from DUC2007 task. From these sets, we collected $\backsim$80K and $\backsim$25K sentences to compile training and testing data respectively. The task is to create max. 250 word long summary for each document cluster.

We use Gibbs sampling for inference in hLDA and sumHLDA. The hLDA is used to capture abstraction and specificity of words in documents (Blei et al., 2009). Contrary to typical hLDA models, to efficiently represent sentences in summarization task, we set ascending values for Dirichlet hyper-parameter η as the level increases, encouraging mid to low level distributions to generate as many words as in higher levels, e.g., for a tree of depth=3, $\eta = \{0.125, 0.5, 1\}$. This causes sentences share paths only when they include similar concepts, starting higher level topics of the tree. For SVR, we set $\epsilon = 0.1$ using the default choice, which is the inverse of the average of $\phi(\mathbf{f})^T \phi(\mathbf{f})$ (Joachims, 1999), dot product of kernelized input vectors. We use greedy optimization during training based on ROUGE scores to find best regularizer $C = \{10^{-1}..10^{2}\}$ using the Gaussian kernel.

We applied feature extraction of § 5.1 to compile the training and testing datasets. ROUGE is used for performance measure (Lin and Hovy, 2003; Lin, 2004), which evaluates summaries based on the maxium number of overlapping units between generated summary text and a set of human summaries. We use R-1 (recall against unigrams), R-2 (recall against bigrams), and R-SU4 (recall against skip-4 bigrams).

Experiment 1: sumHLDA Parameter Analysis: In sumHLDA we introduce a prior different than the standard nested CRP (nCRP). Here, we illustrate that this prior is practical in learning hierarchical topics for summarization task.

We use sentences from the human generated summaries during the discovery of hierarchical topics of sentences in document clusters. Since summary sentences generally contain abstract words, they are indicative of sentences in documents and should produce minimal amount of new topics (if not none). To implement this, in nCRP prior of sumHLDA, we use dual hyper-parameters and choose a very small value for summary sentences, $\gamma_s = 10e^{-4} \ll \gamma_o$. We compare the results to hLDA (Blei et al., 2003a) with nCRP prior which uses only one free parameter, γ. To analyze this prior, we generate a corpus of $\backsim$1300 sentences of a document cluster in DUC2005. We repeated the experiment for 9 other clusters of similar size and averaged the total number of generated topics. We show results for different values of γ and γ_o hyper-parameters and tree depths.

$\gamma = \gamma_o$	**0.1**			**1**			**10**		
depth	**3**	**5**	**8**	**3**	**5**	**8**	**3**	**5**	**8**
hLDA	3	5	8	41	267	1509	1522	4080	8015
sumHLDA	3	5	8	27	162	671	1207	3598	7050

Table 1: Average # of topics per document cluster from sumHLDA and hLDA for different γ and γ_o and tree depths. $\gamma_s = 10e^{-4}$ is used for sumHLDA for each depth.

Features	Baseline			HybHSum		
	R-1	R-2	R-SU4	R-1	R-2	R-SU4
NMF (1)	40.3	7.8	13.7	41.6	8.4	12.3
DMF (2)	41.3	7.5	14.3	41.3	8.0	13.9
OF (3)	40.3	7.4	13.7	**42.4**	8.0	14.4
(1+2)	41.5	7.9	14.0	41.8	8.5	14.5
(1+3)	40.8	7.5	13.8	41.6	8.2	14.1
(2+3)	40.7	7.4	13.8	**42.7**	**8.7**	**14.9**
(1+2+3)	41.4	8.1	13.7	**43.0**	9.1	**15.1**

Table 2: ROUGE results (with stop-words) on DUC2006 for different features and methods. Results in bold show statistical significance over baseline in corresponding metric.

As shown in Table 1, the nCRP prior for sumHLDA is more effective than hLDA prior in the summarization task. Less number of topics(nodes) in sumHLDA suggests that summary sentences share pre-existing paths and no new paths or nodes are sampled for them. We also observe that using $\gamma_o = 0.1$ causes the model to generate minimum number of topics (# of topics=depth), while setting $\gamma_o = 10$ creates excessive amount of topics. $\gamma_0 = 1$ gives reasonable number of topics, thus we use this value for the rest of the experiments. In experiment 3, we use both nCRP priors in HybHSum to analyze whether there is any performance gain with the new prior.

Experiment 2: Feature Selection Analysis

Here we test individual contribution of each set of features on our HybHSum (using sumHLDA). We use a **Baseline** by replacing the scoring algorithm of HybHSum with a simple cosine distance measure. The score of a candidate sentence is the cosine similarity to the maximum matching summary sentence. Later, we build a regression model with the same features as our HybHSum to create a summary. We train models with DUC2005 and evaluate performance on DUC2006 documents for different parameter values as shown in Table 2.

As presented in § 5, NMF is the bundle of frequency based meta-features on document cluster level, DMF is a bundle of frequency based meta-features on individual document level and OF represents sentence term frequency, location, and size features. In comparison to the baseline, OF has a significant effect on the ROUGE scores. In addition, DMF together with OF has shown to improve all scores, in comparison to baseline, on average by 10%. Although the NMF have minimal individual improvement, all these features can statistically improve R-2 without stop words by 12% (significance is measured by t-test statistics).

Experiment 3: ROUGE Evaluations

We use the following multi-document summarization models along with the Baseline presented in Experiment 2 to evaluate HybSumm.

⋆ **PYTHY :** (Toutanova et al., 2007) A state-of-the-art supervised summarization system that ranked first in overall ROUGE evaluations in DUC2007. Similar to HybHSum, human generated summaries are used to train a sentence ranking system using a classifier model.

⋆ **HIERSUM :** (Haghighi and Vanderwende, 2009) A generative summarization method based on topic models, which uses sentences as an additional level. Using an approximation for inference, sentences are greedily added to a summary so long as they decrease KL-divergence.

⋆ **HybFSum (Hybrid Flat Summarizer):** To investigate the performance of hierarchical topic model, we build another hybrid model using flat LDA (Blei et al., 2003b). In LDA each sentence is a superposition of all K topics with sentence specific weights, there is no hierarchical relation between topics. We keep the parameters and the features of the regression model of hierarchical HybHSum intact for consistency. We only change the sentence scoring method. Instead of the new tree-based sentence scoring (§ 4), we present a similar method using topics from LDA on sentence level. Note that in LDA the topic-word distributions ϕ are over entire vocabulary, and topic mixing proportions for sentences θ are over all the topics discovered from sentences in a document cluster. Hence, we define sim_1 and sim_2 measures for LDA using topic-word proportions ϕ (in place of discrete topic-word distributions from each level in Eq.2) and topic mixing weights θ in sentences (in place of topic proportions in Eq.3) respectively. Maximum matching score is calculated as same as in HybHSum.

⋆ **HybHSum**$_1$ and **HybHSum**$_2$: To analyze the effect of the new nCRP prior of sumHLDA on sum-

ROUGE	w/o stop words			w/ stop words		
	R-1	R-2	R-4	R-1	R-2	R-4
Baseline	32.4	7.4	10.6	41.0	9.3	15.2
PYTHY	35.7	8.9	**12.1**	42.6	11.9	16.8
HIERSUM	33.8	**9.3**	11.6	42.4	11.8	16.7
HybFSum	34.5	8.6	10.9	43.6	9.5	15.7
HybHSum_1	34.0	7.9	11.5	44.8	11.0	16.7
HybHSum_2	35.1	8.3	11.8	**45.6**	11.4	**17.2**

Table 3: ROUGE results of the best systems on DUC2007 dataset (best results are **bolded**.)

marization model performance, we build two different versions of our hybrid model: $\textbf{HybHSum}_1$ using standard hLDA (Blei et al., 2003a) and $\textbf{HybHSum}_2$ using our sumHLDA.

The ROUGE results are shown in Table 3. The HybHSum_2 achieves the best performance on R-1 and R-4 and comparable on R-2. When stop words are used the HybHSum_2 outperforms state-of-the-art by 2.5-7% except R-2 (with statistical significance). Note that R-2 is a measure of bigram recall and sumHLDA of HybHSum_2 is built on unigrams rather than bigrams. Compared to the HybFSum built on LDA, both $\text{HybHSum}_{1\&2}$ yield better performance indicating the effectiveness of using hierarchical topic model in summarization task. HybHSum_2 appear to be less redundant than HybFSum capturing not only common terms but also specific words in Fig. 2, due to the new hierarchical tree-based sentence scoring which characterizes sentences on deeper level. Similarly, $\text{HybHSum}_{1\&2}$ far exceeds baseline built on simple classifier. The results justify the performance gain by using our novel tree-based scoring method. Although the ROUGE scores for HybHSum_1 and HybHSum_2 are not significantly different, the sumHLDA is more suitable for summarization tasks than hLDA.

HybHSum_2 is comparable to (if not better than) fully generative HIERSUM. This indicates that with our regression model built on training data, summaries can be efficiently generated for test documents (suitable for online systems).

Experiment 4: Manual Evaluations

Here, we manually evaluate quality of summaries, a common DUC task. Human annotators are given two sets of summary text for each document set, generated from two approaches: best hierarchical hybrid HybHSum_2 and flat hybrid HybFSum models, and are asked to mark the better summary

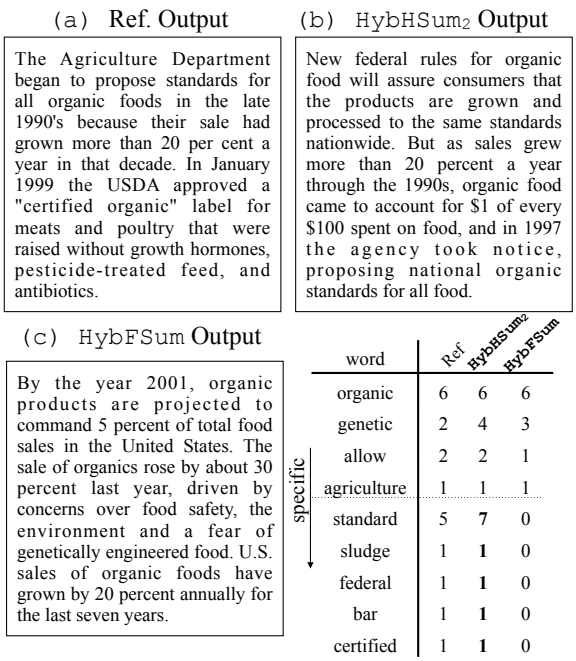


Figure 2: Example summary text generated by systems compared in Experiment 3. (Id:D0744 in DUC2007). Ref. is the human generated summary.

Criteria	HybFSum	HybHSum_2	Tie
Non-redundancy	26	44	22
Coherence	24	56	12
Focus	24	56	12
Responsiveness	30	50	12
Overall	24	66	2

Table 4: Frequency results of manual quality evaluations. Results are statistically significant based on t-test. Tie indicates evaluations where two summaries are rated equal.

according to five criteria: *non-redundancy* (which summary is less redundant), *coherence* (which summary is more coherent), *focus and readability* (content and not include unnecessary details), *responsiveness* and *overall* performance.

We asked 4 annotators to rate DUC2007 predicted summaries (45 summary pairs per annotator). A total of 92 pairs are judged and evaluation results in frequencies are shown in Table 4. The participants rated HybHSum_2 generated summaries more coherent and focused compared to HybFSum. All results in Table 4 are statistically significant (based on t-test on 95% confidence level.) indicating that HybHSum_2 summaries are rated significantly better.

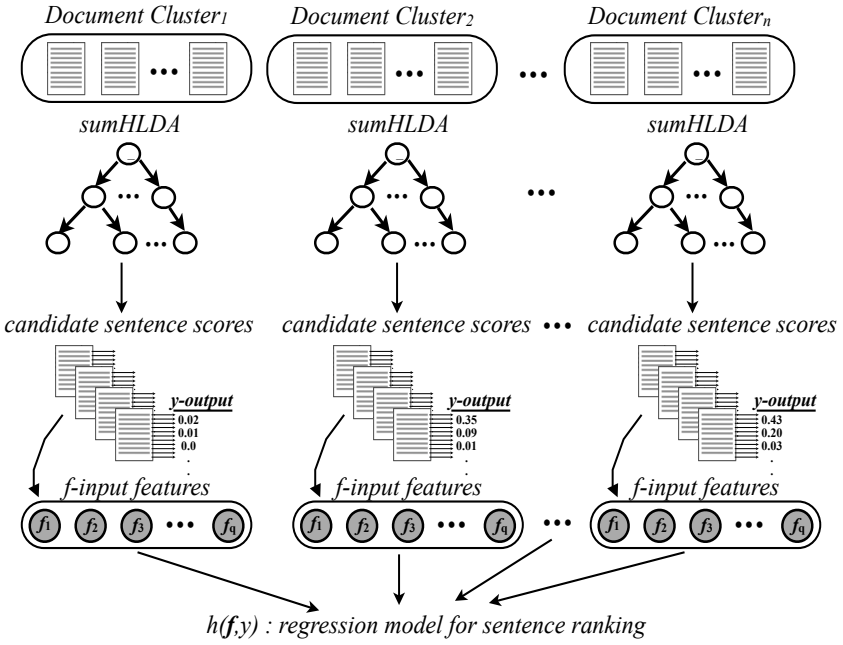


Figure 3: Flow diagram for Hybrid Learning Algorithm for Multi-Document Summarization.

7 Conclusion

In this paper, we presented a hybrid model for multi-document summarization. We demonstrated that implementation of a summary focused hierarchical topic model to discover sentence structures as well as construction of a discriminative method for inference can benefit summarization quality on manual and automatic evaluation metrics.

Acknowledgement

Research supported in part by ONR N00014-02-1-0294, BT Grant CT1080028046, Azerbaijan Ministry of Communications and Information Technology Grant, Azerbaijan University of Azerbaijan Republic and the BISC Program of UC Berkeley.

References

R. Barzilay and L. Lee. Catching the drift: Probabilistic content models with applications to generation and summarization. In *In Proc. HLT-NAACL'04*, 2004.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *In Neural Information Processing Systems [NIPS]*, 2003a.

D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. In *Journal of ACM*, 2009.

D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. In *Jrnl. Machine Learning Research, 3:993-1022*, 2003b.

S.R.K. Branavan, H. Chen, J. Eisenstein, and R. Barzilay. Learning document-level semantic properties from free-text annotations. In *Journal of Artificial Intelligence Research*, volume 34, 2009.

J.M. Conroy, J.D. Schlesinger, and D.P. O'Leary. Topic focused multi-cument summarization using an approximate oracle score. In *In Proc. ACL'06*, 2006.

H. Daumé III and D. Marcu. Bayesian query focused summarization. In *Proc. ACL-06*, 2006.

H. Drucker, C.J.C. Burger, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *NIPS 9*, 1997.

A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *NAACL HLT-09*, 2009.

T. Joachims. Making large-scale svm learning practical. In *In Advances in Kernel Methods - Support Vector Learning. MIT Press.*, 1999.

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *In Proc. ACL Workshop on Text Summarization Branches Out*, 2004.

C.-Y. Lin and E.H. Hovy. Automatic evaluation of summaries using n-gram co-occurance statistics. In *Proc. HLT-NAACL, Edmonton, Canada*, 2003.

C. Manning and H. Schuetze. Foundations of statistical natural language processing. In *MIT Press. Cambridge, MA*, 1999.

A. Nenkova and L. Vanderwende. The impact of frequency on summarization. In *Tech. Report MSR-TR-2005-101, Microsoft Research, Redwood, Washington*, 2005.

D.R. Radev, H. Jing, M. Stys, and D. Tam. Centroid-based summarization for multiple documents. In *In Int. Jrnl. Information Processing and Management*, 2004.

D. Shen, J.T. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random fields. In *Proc. IJCAI'07*, 2007.

J. Tang, L. Yao, and D. Chens. Multi-topic based query-oriented summarization. In *SIAM International Conference Data Mining*, 2009.

I. Titov and R. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL-08:HLT*, 2008.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. The phthy summarization system: Microsoft research at duc 2007. In *Proc. DUC*, 2007.

J.Y. Yeh, H.-R. Ke, W.P. Yang, and I-H. Meng. Text summarization using a trainable summarizer and latent semantic analysis. In *Information Processing and Management*, 2005.

Improving Statistical Machine Translation with Monolingual Collocation

Zhanyi Liu[1], Haifeng Wang[2], Hua Wu[2], Sheng Li[1]
[1]Harbin Institute of Technology, Harbin, China
[2]Baidu.com Inc., Beijing, China
zhanyiliu@gmail.com
{wanghaifeng, wu_hua}@baidu.com
lisheng@hit.edu.cn

Abstract

This paper proposes to use monolingual collocations to improve Statistical Machine Translation (SMT). We make use of the collocation probabilities, which are estimated from monolingual corpora, in two aspects, namely improving word alignment for various kinds of SMT systems and improving phrase table for phrase-based SMT. The experimental results show that our method improves the performance of both word alignment and translation quality significantly. As compared to baseline systems, we achieve absolute improvements of 2.40 BLEU score on a phrase-based SMT system and 1.76 BLEU score on a parsing-based SMT system.

1 Introduction

Statistical bilingual word alignment (Brown et al. 1993) is the base of most SMT systems. As compared to single-word alignment, multi-word alignment is more difficult to be identified. Although many methods were proposed to improve the quality of word alignments (Wu, 1997; Och and Ney, 2000; Marcu and Wong, 2002; Cherry and Lin, 2003; Liu et al., 2005; Huang, 2009), the correlation of the words in multi-word alignments is not fully considered.

In phrase-based SMT (Koehn et al., 2003), the phrase boundary is usually determined based on the bi-directional word alignments. But as far as we know, few previous studies exploit the collocation relations of the words in a phrase. Some researches used soft syntactic constraints to predict whether source phrase can be translated together (Marton and Resnik, 2008; Xiong et al., 2009). However, the constraints were learned from the parsed corpus, which is not available for many languages.

In this paper, we propose to use monolingual collocations to improve SMT. We first identify potentially collocated words and estimate collocation probabilities from monolingual corpora using a Monolingual Word Alignment (MWA) method (Liu et al., 2009), which does not need any additional resource or linguistic preprocessing, and which outperforms previous methods on the same experimental data. Then the collocation information is employed to improve Bilingual Word Alignment (BWA) for various kinds of SMT systems and to improve phrase table for phrase-based SMT.

To improve BWA, we re-estimate the alignment probabilities by using the collocation probabilities of words in the same cept. A cept is the set of source words that are connected to the same target word (Brown et al., 1993). An alignment between a source multi-word cept and a target word is a many-to-one multi-word alignment.

To improve phrase table, we calculate phrase collocation probabilities based on word collocation probabilities. Then the phrase collocation probabilities are used as additional features in phrase-based SMT systems.

The evaluation results show that the proposed method in this paper significantly improves multi-word alignment, achieving an absolute error rate reduction of 29%. The alignment improvement results in an improvement of 2.16 BLEU score on phrase-based SMT system and an improvement of 1.76 BLEU score on parsing-based SMT system. If we use phrase collocation probabilities as additional features, the phrase-based

This work was partially done at Toshiba (China) Research and Development Center.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 825–833,
Uppsala, Sweden, 11-16 July 2010.

SMT performance is further improved by 0.24 BLEU score.

The paper is organized as follows: In section 2, we introduce the collocation model based on the MWA method. In section 3 and 4, we show how to improve the BWA method and the phrase table using collocation models respectively. We describe the experimental results in section 5, 6 and 7. Lastly, we conclude in section 8.

2 Collocation Model

Collocation is generally defined as a group of words that occur together more often than by chance (McKeown and Radev, 2000). A collocation is composed of two words occurring as either a consecutive word sequence or an interrupted word sequence in sentences, such as "by accident" or "take ... advice". In this paper, we use the MWA method (Liu et al., 2009) for collocation extraction. This method adapts the bilingual word alignment algorithm to monolingual scenario to extract collocations only from monolingual corpora. And the experimental results in (Liu et al., 2009) showed that this method achieved higher precision and recall than previous methods on the same experimental data.

2.1 Monolingual word alignment

The monolingual corpus is first replicated to generate a parallel corpus, where each sentence pair consists of two identical sentences in the same language. Then the monolingual word alignment algorithm is employed to align the potentially collocated words in the monolingual sentences.

According to Liu et al. (2009), we employ the MWA Model 3 (corresponding to IBM Model 3) to calculate the probability of the monolingual word alignment sequence, as shown in Eq. (1).

$$p_{\mathrm{MWAModel3}}(S, A \mid S) \propto \prod_{i=1}^{l} n(\phi_i \mid w_i) \cdot \prod_{j=1}^{l} t(w_j \mid w_{a_j}) \cdot d(j \mid a_j, l) \quad (1)$$

Where $S = w_1^l$ is a monolingual sentence, ϕ_i denotes the number of words that are aligned with w_i. Since a word never collocates with itself, the alignment set is denoted as $A = \{(i, a_i) \mid i \in [1, l] \,\&\, a_i \neq i\}$. Three kinds of probabilities are involved in this model: word collocation probability $t(w_j \mid w_{a_j})$, position collocation probability $d(j \mid a_j, l)$ and fertility probability $n(\phi_i \mid w_i)$.

In the MWA method, the similar algorithm to bilingual word alignment is used to estimate the parameters of the models, except that a word cannot be aligned to itself.

Figure 1 shows an example of the potentially collocated word pairs aligned by the MWA method.

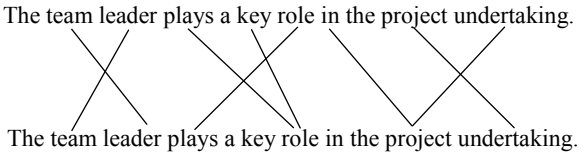


Figure 1. MWA Example

2.2 Collocation probability

Given the monolingual word aligned corpus, we calculate the frequency of two words aligned in the corpus, denoted as $freq(w_i, w_j)$. We filtered the aligned words occurring only once. Then the probability for each aligned word pair is estimated as follows:

$$p(w_i \mid w_j) = \frac{freq(w_i, w_j)}{\sum_{w'} freq(w', w_j)} \quad (2)$$

$$p(w_j \mid w_i) = \frac{freq(w_i, w_j)}{\sum_{w'} freq(w_i, w')} \quad (3)$$

In this paper, the words of collocation are symmetric and we do not determine which word is the head and which word is the modifier. Thus, the collocation probability of two words is defined as the average of both probabilities, as in Eq. (4).

$$r(w_i, w_j) = \frac{p(w_i \mid w_j) + p(w_j \mid w_i)}{2} \quad (4)$$

If we have multiple monolingual corpora to estimate the collocation probabilities, we interpolate the probabilities as shown in Eq. (5).

$$r(w_i, w_j) = \sum_k \alpha_k r_k(w_i, w_j) \quad (5)$$

α_k denotes the interpolation coefficient for the probabilities estimated on the k^{th} corpus.

3 Improving Statistical Bilingual Word Alignment

We use the collocation information to improve both one-directional and bi-directional bilingual word alignments. The alignment probabilities are re-estimated by using the collocation probabilities of words in the same cept.

3.1 Improving one-directional bilingual word alignment

According to the BWA method, given a bilingual sentence pair $E = e_1^l$ and $F = f_1^m$, the optimal alignment sequence A between E and F can be obtained as in Eq. (6).

$$A^* = \arg\max_A p(F, A \mid E) \quad (6)$$

The method is implemented in a series of five models (IBM Models). IBM Model 1 only employs the word translation model to calculate the probabilities of alignments. In IBM Model 2, both the word translation model and position distribution model are used. IBM Model 3, 4 and 5 consider the fertility model in addition to the word translation model and position distribution model. And these three models are similar, except for the word distortion models.

One-to-one and many-to-one alignments could be produced by using IBM models. Although the fertility model is used to restrict the number of source words in a cept and the position distortion model is used to describe the correlation of the positions of the source words, the quality of many-to-one alignments is lower than that of one-to-one alignments.

Intuitively, the probability of the source words aligned to a target word is not only related to the fertility ability and their relative positions, but also related to lexical tokens of words, such as common phrase or idiom. In this paper, we use the collocation probability of the source words in a cept to measure their correlation strength. Given source words $\{f_j \mid a_j = i\}$ aligned to e_i, their collocation probability is calculated as in Eq. (7).

$$r(\{f_j \mid a_j = i\}) = \frac{2\sum_{k=1}^{\phi_i - 1}\sum_{g=k+1}^{\phi_i} r(f_{[i]k}, f_{[i]g})}{\phi_i * (\phi_i - 1)} \quad (7)$$

Here, $f_{[i]k}$ and $f_{[i]g}$ denote the k^{th} word and g^{th} word in $\{f_j \mid a_j = i\}$; $r(f_{[i]k}, f_{[i]g})$ denotes the collocation probability of $f_{[i]k}$ and $f_{[i]g}$, as shown in Eq. (4).

Thus, the collocation probability of the alignment sequence of a sentence pair can be calculated according to Eq. (8).

$$r(F, A \mid E) = \prod_{i=1}^{l} r(\{f_j \mid a_j = i\}) \quad (8)$$

Based on maximum entropy framework, we combine the collocation model and the BWA model to calculate the word alignment probability of a sentence pair, as shown in Eq. (9).

$$p_r(F, A \mid E) = \frac{\exp(\sum_i \lambda_i h_i(F, E, A))}{\sum_{A'} \exp(\sum_i \lambda_i h_i(F, E, A'))} \quad (9)$$

Here, $h_i(F, E, A)$ and λ_i denote features and feature weights, respectively. We use two features in this paper, namely alignment probabilities and collocation probabilities.

Thus, we obtain the decision rule:

$$A^* = \arg\max_A \{\sum_i \lambda_i h_i(F, E, A)\} \quad (10)$$

Based on the GIZA++ package[1], we implemented a tool for the improved BWA method. We first train IBM Model 4 and collocation model on bilingual corpus and monolingual corpus respectively. Then we employ the hill-climbing algorithm (Al-Onaizan et al., 1999) to search for the optimal alignment sequence of a given sentence pair, where the score of an alignment sequence is calculated as in Eq. (10).

We note that Eq. (8) only deals with many-to-one alignments, but the alignment sequence of a sentence pair also includes one-to-one alignments. To calculate the collocation probability of the alignment sequence, we should also consider the collocation probabilities of such one-to-one alignments. To solve this problem, we use the collocation probability of the whole source sentence, $r(F)$, as the collocation probability of one-word cept.

3.2 Improving bi-directional bilingual word alignments

In word alignment models implemented in GIZA++, only one-to-one and many-to-one word alignment links can be found. Thus, some multi-word units cannot be correctly aligned. The symmetrization method is used to effectively overcome this deficiency (Och and Ney, 2003). Bi-directional alignments are generally obtained from source-to-target alignments A_{s2t} and target-to-source alignments A_{t2s}, using some heuristic rules (Koehn et al., 2005). This method ignores the correlation of the words in the same alignment unit, so an alignment may include many unrelated words[2], which influences the performances of SMT systems.

[1] http://www.fjoch.com/GIZA++.html

[2] In our experiments, a multi-word unit may include up to 40 words.

In order to solve the above problem, we incorporate the collocation probabilities into the bi-directional word alignment process.

Given alignment sets A_{s2t} and A_{t2s}. We can obtain the union $A_{s\leftrightarrow t} = A_{s2t} \bigcup A_{t2s}$. The source sentence f_1^m can be segmented into m' cepts $\bar{f}_1^{m'}$. The target sentence e_1^l can also be segmented into l' cepts $\bar{e}_1^{l'}$. The words in the same cept can be a consecutive word sequence or an interrupted word sequence.

Finally, the optimal alignments $\bar{A}$ between $\bar{f}_1^{m'}$ and $\bar{e}_1^{l'}$ can be obtained from $A_{s\leftrightarrow t}$ using the following decision rule.

$$(\bar{e}_1^{l'}, \bar{f}_1^{m'}, \bar{A})^* = \arg\max_{\bar{A}\subseteq A_{s\leftrightarrow t}} \{ \prod_{(\bar{e}_i,\bar{f}_j)\in\bar{A}} p(\bar{e}_i,\bar{f}_j)^{\tau_1} \cdot r(\bar{e}_i)^{\tau_2} \cdot r(\bar{f}_j)^{\tau_3} \} \quad (11)$$

Here, $r(\bar{f}_j)$ and $r(\bar{e}_i)$ denote the collocation probabilities of the words in the source language and target language respectively, which are calculated by using Eq. (7). $p(\bar{e}_i,\bar{f}_j)$ denotes the word translation probability that is calculated according to Eq. (12). τ_i denotes the weights of these probabilities.

$$p(\bar{e}_i,\bar{f}_j) = \frac{\sum_{e\in\bar{e}_i}\sum_{f\in\bar{f}_j}(p(e|f)+p(f|e))/2}{|\bar{e}_i|*|\bar{f}_j|} \quad (12)$$

$p(e|f)$ and $p(f|e)$ are the source-to-target and target-to-source translation probabilities trained from the word aligned bilingual corpus.

4 Improving Phrase Table

Phrase-based SMT system automatically extracts bilingual phrase pairs from the word aligned bilingual corpus. In such a system, an idiomatic expression may be split into several fragments, and the phrases may include irrelevant words. In this paper, we use the collocation probability to measure the possibility of words composing a phrase.

For each bilingual phrase pair automatically extracted from word aligned corpus, we calculate the collocation probabilities of source phrase and target phrase respectively, according to Eq. (13).

$$r(w_1^n) = \frac{2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} r(w_i,w_j)}{n*(n-1)} \quad (13)$$

Here, w_1^n denotes a phrase with n words; $r(w_i,w_j)$ denotes the collocation probability of a word pair calculated according to Eq. (4). For the phrase only including one word, we set a fixed collocation probability that is the average of the collocation probabilities of the sentences on a development set. These collocation probabilities are incorporated into the phrase-based SMT system as features.

Corpora	Chinese words	English words
Bilingual corpus	6.3M	8.5M
Additional monolingual corpora	312M	203M

Table 1. Statistics of training data

5 Experiments on Word Alignment

5.1 Experimental settings

We use a bilingual corpus, FBIS (LDC2003E14), to train the IBM models. To train the collocation models, besides the monolingual parts of FBIS, we also employ some other larger Chinese and English monolingual corpora, namely, Chinese Gigaword (LDC2007T38), English Gigaword (LDC2007T07), UN corpus (LDC2004E12), Sinorama corpus (LDC2005T10), as shown in Table 1.

Using these corpora, we got three kinds of collocation models:

CM-1: the training data is the additional monolingual corpora;
CM-2: the training data is either side of the bilingual corpus;
CM-3: the interpolation of CM-1 and CM-2.

To investigate the quality of the generated word alignments, we randomly selected a subset from the bilingual corpus as test set, including 500 sentence pairs. Then word alignments in the subset were manually labeled, referring to the guideline of the Chinese-to-English alignment (LDC2006E93), but we made some modifications for the guideline. For example, if a preposition appears after a verb as a phrase aligned to one single word in the corresponding sentence, then they are glued together.

There are several different evaluation metrics for word alignment (Ahrenberg et al., 2000). We use precision (P), recall (R) and alignment error ratio (AER), which are similar to those in Och and Ney (2000), except that we consider each alignment as a sure link.

Experiments		Single word alignments			Multi-word alignments		
		P	R	AER	P	R	AER
Baseline		0.77	0.45	0.43	0.23	0.71	0.65
Improved BWA methods	CM-1	0.70	0.50	0.42	0.35	0.86	0.50
	CM-2	0.73	0.48	0.42	0.36	0.89	0.49
	CM-3	0.73	0.48	0.41	0.39	0.78	0.47

Table 2. English-to-Chinese word alignment results

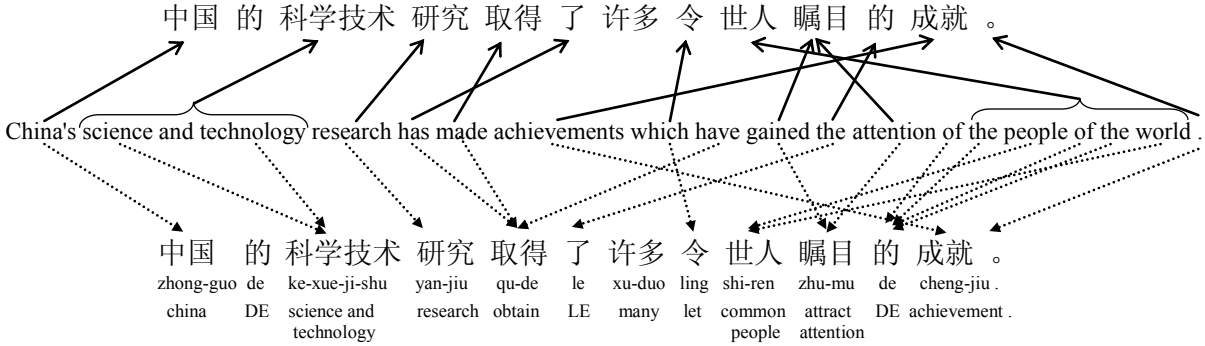


Figure 2. Example of the English-to-Chinese word alignments generated by the BWA method and the improved BWA method using CM-3. "⟶" denotes the alignments of our method; "⋯→" denotes the alignments of the baseline method.

$$P = \frac{|S_g \cap S_r|}{|S_g|} \tag{14}$$

$$R = \frac{|S_g \cap S_r|}{|S_r|} \tag{15}$$

$$AER = 1 - \frac{2*|S_g \cap S_r|}{|S_g| + |S_r|} \tag{16}$$

Where, S_g and S_r denote the automatically generated alignments and the reference alignments.

In order to tune the interpolation coefficients in Eq. (5) and the weights of the probabilities in Eq. (11), we also manually labeled a development set including 100 sentence pairs, in the same manner as the test set. By minimizing the AER on the development set, the interpolation coefficients of the collocation probabilities on CM-1 and CM-2 were set to 0.1 and 0.9. And the weights of probabilities were set as $\tau_1 = 0.6$, $\tau_2 = 0.2$ and $\tau_3 = 0.2$.

5.2 Evaluation results

One-directional alignment results

To train a Chinese-to-English SMT system, we need to perform both Chinese-to-English and English-to-Chinese word alignment. We only evaluate the English-to-Chinese word alignment here. GIZA++ with the default settings is used as the baseline method. The evaluation results in Table 2 indicate that the performances of our methods on single word alignments are close to that of the baseline method. For multi-word alignments, our methods significantly outperform the baseline method in terms of both precision and recall, achieving up to 18% absolute error rate reduction.

Although the size of the bilingual corpus is much smaller than that of additional monolingual corpora, our methods using CM-1 and CM-2 achieve comparable performances. It is because CM-2 and the BWA model are derived from the same resource. By interpolating CM1 and CM2, i.e. CM-3, the error rate of multi-word alignment results is further reduced.

Figure 2 shows an example of word alignment results generated by the baseline method and the improved method using CM-3. In this example, our method successfully identifies many-to-one alignments such as "the people of the world ⟶ 世人". In our collocation model, the collocation probability of "the people of the world" is much higher than that of "people world". And our method is also effective to prevent the unrelated

Experiments		Single word alignments			Multi-word alignments			All alignments		
		P	R	AER	P	R	AER	P	R	AER
Baseline		0.84	0.43	0.42	0.18	0.74	0.70	0.52	0.45	0.51
Our methods	WA-1	0.80	0.51	0.37	0.30	0.89	0.55	0.58	0.51	0.45
	WA-2	0.81	0.50	0.37	0.33	0.81	0.52	0.62	0.50	0.44
	WA-3	0.78	0.56	0.34	0.44	0.88	0.41	0.63	0.54	0.40

Table 3. Bi-directional word alignment results

words from being aligned. For example, in the baseline alignment "has made ... have ⋯▸取得", "have" and "has" are unrelated to the target word, while our method only generated "made → 取得", this is because that the collocation probabilities of "has/have" and "made" are much lower than that of the whole source sentence.

Bi-directional alignment results

We build a bi-directional alignment baseline in two steps: (1) GIZA++ is used to obtain the source-to-target and target-to-source alignments; (2) the bi-directional alignments are generated by using "grow-diag-final". We use the methods proposed in section 3 to replace the corresponding steps in the baseline method. We evaluate three methods:

WA-1: one-directional alignment method proposed in section 3.1 and grow-diag-final;
WA-2: GIZA++ and the bi-directional bilingual word alignments method proposed in section 3.2;
WA-3: both methods proposed in section 3.

Here, CM-3 is used in our methods. The results are shown in Table 3.

We can see that WA-1 achieves lower alignment error rate as compared to the baseline method, since the performance of the improved one-directional alignment method is better than that of GIZA++. This result indicates that improving one-directional word alignment results in bi-directional word alignment improvement.

The results also show that the AER of WA-2 is lower than that of the baseline. This is because the proposed bi-directional alignment method can effectively recognize the correct alignments from the alignment union, by leveraging collocation probabilities of the words in the same cept.

Our method using both methods proposed in section 3 produces the best alignment performance, achieving 11% absolute error rate reduction.

Experiments			BLEU (%)
Baseline			29.62
Our methods	WA-1	CM-1	30.85
		CM-2	31.28
		CM-3	31.48
	WA-2	CM-1	31.00
		CM-2	31.33
		CM-3	31.51
	WA-3	CM-1	31.43
		CM-2	31.62
		CM-3	31.78

Table 4. Performances of Moses using the different bi-directional word alignments (Significantly better than baseline with $p < 0.01$)

6 Experiments on Phrase-Based SMT

6.1 Experimental settings

We use FBIS corpus to train the Chinese-to-English SMT systems. Moses (Koehn et al., 2007) is used as the baseline phrase-based SMT system. We use SRI language modeling toolkit (Stolcke, 2002) to train a 5-gram language model on the English sentences of FBIS corpus. We used the NIST MT-2002 set as the development set and the NIST MT-2004 test set as the test set. And Koehn's implementation of minimum error rate training (Och, 2003) is used to tune the feature weights on the development set.

We use BLEU (Papineni et al., 2002) as evaluation metrics. We also calculate the statistical significance differences between our methods and the baseline method by using paired bootstrap re-sample method (Koehn, 2004).

6.2 Effect of improved word alignment on phrase-based SMT

We investigate the effectiveness of the improved word alignments on the phrase-based SMT system. The bi-directional alignments are obtained

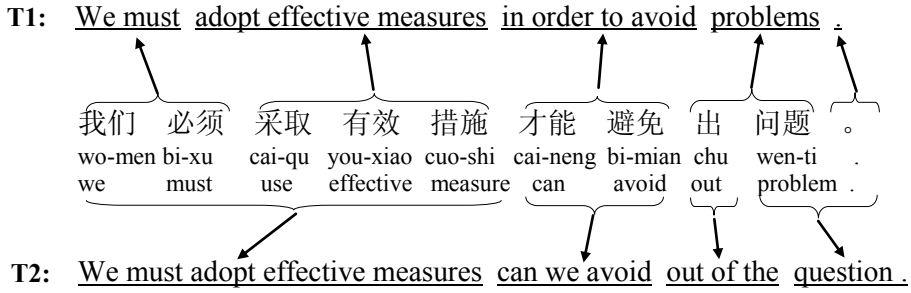


Figure 3. Example of the translations generated by the baseline system and the system where the phrase collocation probabilities are added

Experiments	BLEU (%)
Moses	29.62
+ Phrase collocation probability	30.47
+ Improved word alignments + Phrase collocation probability	32.02

Table 5. Performances of Moses employing our proposed methods (Significantly better than baseline with $p < 0.01$)

using the same methods as those shown in Table 3. Here, we investigate three different collocation models for translation quality improvement. The results are shown in Table 4.

From the results of Table 4, it can be seen that the systems using the improved bi-directional alignments achieve higher quality of translation than the baseline system. If the same alignment method is used, the systems using CM-3 got the highest BLEU scores. And if the same collocation model is used, the systems using WA-3 achieved the higher scores. These results are consistent with the evaluations of word alignments as shown in Tables 2 and 3.

6.3 Effect of phrase collocation probabilities

To investigate the effectiveness of the method proposed in section 4, we only use the collocation model CM-3 as described in section 5.1. The results are shown in Table 5. When the phrase collocation probabilities are incorporated into the SMT system, the translation quality is improved, achieving an absolute improvement of 0.85 BLEU score. This result indicates that the collocation probabilities of phrases are useful in determining the boundary of phrase and predicting whether phrases should be translated together, which helps to improve the phrase-based SMT performance.

Figure 3 shows an example: T1 is generated by the system where the phrase collocation probabilities are used and T2 is generated by the baseline system. In this example, since the collocation probability of "出 问题" is much higher than that of "问题 。", our method tends to split "出 问题 。" into "(出 问题) (。)", rather than "(出) (问题 。)". For the phrase "才能 避免" in the source sentence, the collocation probability of the translation "in order to avoid" is higher than that of the translation "can we avoid". Thus, our method selects the former as the translation. Although the phrase "我们 必须 采取 有效 措施" in the source sentence has the same translation "We must adopt effective measures", our method splits this phrase into two parts "我们 必须" and "采取 有效 措施", because two parts have higher collocation probabilities than the whole phrase.

We also investigate the performance of the system employing both the word alignment improvement and phrase table improvement methods. From the results in Table 5, it can be seen that the quality of translation is future improved. As compared with the baseline system, an absolute improvement of 2.40 BLEU score is achieved. And this result is also better than the results shown in Table 4.

7 Experiments on Parsing-Based SMT

We also investigate the effectiveness of the improved word alignments on the parsing-based SMT system, Joshua (Li et al., 2009). In this system, the Hiero-style SCFG model is used (Chiang, 2007), without syntactic information. The rules are extracted only based on the FBIS corpus, where words are aligned by "MW-3 & CM-3". And the language model is the same as that in Moses. The feature weights are tuned on the development set using the minimum error

Experiments	BLEU (%)
Joshua	30.05
+ Improved word alignments	31.81

Table 6. Performances of Joshua using the different word alignments (Significantly better than baseline with $p < 0.01$)

rate training method. We use the same evaluation measure as described in section 6.1.

The translation results on Joshua are shown in Table 6. The system using the improved word alignments achieves an absolute improvement of 1.76 BLEU score, which indicates that the improvements of word alignments are also effective to improve the performance of the parsing-based SMT systems.

8 Conclusion

We presented a novel method to use monolingual collocations to improve SMT. We first used the MWA method to identify potentially collocated words and estimate collocation probabilities only from monolingual corpora, no additional resource or linguistic preprocessing is needed. Then the collocation information was employed to improve BWA for various kinds of SMT systems and to improve phrase table for phrase-based SMT.

To improve BWA, we re-estimate the alignment probabilities by using the collocation probabilities of words in the same cept. To improve phrase table, we calculate phrase collocation probabilities based on word collocation probabilities. Then the phrase collocation probabilities are used as additional features in phrase-based SMT systems.

The evaluation results showed that the proposed method significantly improved word alignment, achieving an absolute error rate reduction of 29% on multi-word alignment. The improved word alignment results in an improvement of 2.16 BLEU score on a phrase-based SMT system and an improvement of 1.76 BLEU score on a parsing-based SMT system. When we also used phrase collocation probabilities as additional features, the phrase-based SMT performance is finally improved by 2.40 BLEU score as compared with the baseline system.

Reference

Lars Ahrenberg, Magnus Merkel, Anna Sagvall Hein, and Jorg Tiedemann. 2000. Evaluation of Word Alignment Systems. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pp. 1255-1261.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical Machine Translation. Final Report. In *Johns Hopkins University Workshop*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2): 263-311.

Colin Cherry and Dekang Lin. 2003. A Probability Model to Improve Word Alignment. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 88-95.

David Chiang. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2): 201-228.

Fei Huang. 2009. Confidence Measure for Word Alignment. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP*, pp. 932-940.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388-395.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Processings of the International Workshop on Spoken Language Translation 2005*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics*, pp. 127-133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL, Poster and Demonstration Sessions*, pp. 177-180.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Demonstration of Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In *Proceedings of the 47th Annual Meeting of the As-*

sociation for Computational Linguistics, Software Demonstrations, pp. 25-28.

Yang Liu, Qun Liu, and Shouxun Lin. Log-linear Models for Word Alignment. 2005. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 459-466.

Zhanyi Liu, Haifeng Wang, Hua Wu, and Sheng Li. 2009. Collocation Extraction Using Monolingual Word Alignment Method. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 487-495.

Daniel Marcu and William Wong. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pp. 133-139.

Yuval Marton and Philip Resnik. 2008. Soft Syntactic Constraints for Hierarchical Phrase-Based Translation. In *Proceedings of the 46st Annual Meeting of the Association for Computational Linguistics*, pp. 1003-1011.

Kathleen R. McKeown and Dragomir R. Radev. 2000. Collocations. In Robert Dale, Hermann Moisl, and Harold Somers (Ed.), *A Handbook of Natural Language Processing*, pp. 507-523.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 440-447.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 160-167.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1): 19-52.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th annual meeting of the Association for Computational Linguistics*, pp. 311-318.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings for the International Conference on Spoken Language Processing*, pp. 901-904.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3): 377-403.

Deyi Xiong, Min Zhang, Aiti Aw, and Haizhou Li. 2009. A Syntax-Driven Bracketing Model for Phrase-Based Translation. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP*, pp. 315-323.

Bilingual Sense Similarity for Statistical Machine Translation

Boxing Chen, George Foster and **Roland Kuhn**
National Research Council Canada
283 Alexandre-Taché Boulevard, Gatineau (Québec), Canada J8X 3X7
{Boxing.Chen, George.Foster, Roland.Kuhn}@nrc.ca

Abstract

This paper proposes new algorithms to compute the sense similarity between two units (words, phrases, rules, *etc.*) from parallel corpora. The sense similarity scores are computed by using the vector space model. We then apply the algorithms to statistical machine translation by computing the sense similarity between the source and target side of translation rule pairs. Similarity scores are used as additional features of the translation model to improve translation performance. Significant improvements are obtained over a state-of-the-art hierarchical phrase-based machine translation system.

1 Introduction

The sense of a term can generally be inferred from its context. The underlying idea is that a term is characterized by the contexts it co-occurs with. This is also well known as *the Distributional Hypothesis* (Harris, 1954): terms occurring in similar contexts tend to have similar meanings. There has been a lot of work to compute the sense similarity between terms based on their distribution in a corpus, such as (Hindle, 1990; Lund and Burgess, 1996; Landauer and Dumais, 1997; Lin, 1998; Turney, 2001; Pantel and Lin, 2002; Pado and Lapata, 2007).

In the work just cited, a common procedure is followed. Given two terms to be compared, one first extracts various features for each term from their contexts in a corpus and forms a vector space model (VSM); then, one computes their similarity by using similarity functions. The features include words within a surface window of a fixed size (Lund and Burgess, 1996), grammatical dependencies (Lin, 1998; Pantel and Lin 2002; Pado and Lapata, 2007), *etc.* The similarity function which has been most widely used is cosine distance (Salton and McGill, 1983); other similarity functions include Euclidean distance, City Block distance (Bullinaria and Levy; 2007), and Dice and Jaccard coefficients (Frakes and Baeza-Yates, 1992), *etc.* Measures of monolingual sense similarity have been widely used in many applications, such as synonym recognizing (Landauer and Dumais, 1997), word clustering (Pantel and Lin 2002), word sense disambiguation (Yuret and Yatbaz 2009), *etc.*

Use of the vector space model to compute sense similarity has also been adapted to the multilingual condition, based on the assumption that two terms with similar meanings often occur in comparable contexts across languages. Fung (1998) and Rapp (1999) adopted VSM for the application of extracting translation pairs from comparable or even unrelated corpora. The vectors in different languages are first mapped to a common space using an initial bilingual dictionary, and then compared.

However, there is no previous work that uses the VSM to compute sense similarity for terms from parallel corpora. The sense similarities, i.e. the translation probabilities in a translation model, for units from parallel corpora are mainly based on the co-occurrence counts of the two units. Therefore, questions emerge: how good is the sense similarity computed via VSM for two units from parallel corpora? Is it useful for multilingual applications, such as statistical machine translation (SMT)?

In this paper, we try to answer these questions, focusing on sense similarity applied to the SMT task. For this task, translation rules are heuristically extracted from automatically word-aligned sentence pairs. Due to noise in the training corpus or wrong word alignment, the source and target sides of some rules are not semantically equivalent, as can be seen from the following

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 834–843,
Uppsala, Sweden, 11-16 July 2010.

real examples which are taken from the rule table built on our training data (Section 5.1):

世界 上 X 之一 ||| *one of X* (*)
世界 上 X 之一 ||| *one of X in the world*
许多 市民 ||| *many citizens*
许多 市民 ||| *many hong kong residents* (*)

The source and target sides of the rules with (*) at the end are not semantically equivalent; it seems likely that measuring the semantic similarity from their context between the source and target sides of rules might be helpful to machine translation.

In this work, we first propose new algorithms to compute the sense similarity between two units (unit here includes word, phrase, rule, *etc*.) in different languages by using their contexts. Second, we use the sense similarities between the source and target sides of a translation rule to improve statistical machine translation performance.

This work attempts to measure directly the sense similarity for units from different languages by comparing their contexts[1]. Our contribution includes proposing new bilingual sense similarity algorithms and applying them to machine translation.

We chose a hierarchical phrase-based SMT system as our baseline; thus, the units involved in computation of sense similarities are hierarchical rules.

2 Hierarchical phrase-based MT system

The hierarchical phrase-based translation method (Chiang, 2005; Chiang, 2007) is a formal syntax-based translation modeling method; its translation model is a weighted synchronous context free grammar (SCFG). No explicit linguistic syntactic information appears in the model. An SCFG rule has the following form:

$$X \rightarrow \langle \alpha, \gamma, \sim \rangle$$

where X is a non-terminal symbol shared by all the rules; each rule has at most two non-terminals. α (γ) is a source (target) string consisting of terminal and non-terminal symbols. $\sim$ defines a one-to-one correspondence between non-terminals in α and γ.

[1] There has been a lot of work (more details in Section 7) on applying word sense disambiguation (WSD) techniques in SMT for translation selection. However, WSD techniques for SMT do so indirectly, using source-side context to help select a particular translation for a source rule.

	source	target
Ini. phr.	*他 出席 了 会议*	*he attended the meeting*
Rule 1 Context 1	*他 出席 了 X_1* *会议*	*he attended X_1* *the, meeting*
Rule 2 Context 2	*会议* *他, 出席, 了*	*the meeting* *he, attended*
Rule 3 Context 3	*他 X_1 会议* *出席, 了*	*he X_1 the meeting* *attended*
Rule 4 Context 4	*出席 了* *他,会议*	*attended* *he, the, meeting*

Figure 1: example of hierarchical rule pairs and their context features.

Rule frequencies are counted during rule extraction over word-aligned sentence pairs, and they are normalized to estimate features on rules. Following (Chiang, 2005; Chiang, 2007), 4 features are computed for each rule:

- $P(\gamma|\alpha)$ and $P(\alpha|\gamma)$ are direct and inverse rule-based conditional probabilities;
- $P_w(\gamma|\alpha)$ and $P_w(\alpha|\gamma)$ are direct and inverse lexical weights (Koehn et al., 2003).

Empirically, this method has yielded better performance on language pairs such as Chinese-English than the phrase-based method because it permits phrases with gaps; it generalizes the normal phrase-based models in a way that allows long-distance reordering (Chiang, 2005; Chiang, 2007). We use the *Joshua* implementation of the method for decoding (Li et al., 2009).

3 Bag-of-Words Vector Space Model

To compute the sense similarity via VSM, we follow the previous work (Lin, 1998) and represent the source and target side of a rule by feature vectors. In our work, each feature corresponds to a context word which co-occurs with the translation rule.

3.1 Context Features

In the hierarchical phrase-based translation method, the translation rules are extracted by abstracting some words from an initial phrase pair (Chiang, 2005). Consider a rule with non-terminals on the source and target side; for a given instance of the rule (a particular phrase pair in the training corpus), the context will be the words instantiating the non-terminals. In turn, the context for the sub-phrases that instantiate the non-terminals will be the words in the remainder of the phrase pair. For example in Figure 1, if we

have an initial phrase pair *他 出席 了 会议 ||| he attended the meeting*, and we extract four rules from this initial phrase: *他 出席 了 X_1 ||| he attended X_1*, *会议 ||| the meeting*, *他 X_1 会议 ||| he X_1 the meeting*, and *出席 了 ||| attended*. Therefore, *the* and *meeting* are context features of target pattern *he attended X_1*; *he* and *attended* are the context features of *the meeting*; *attended* is the context feature of *he X_1 the meeting*; also *he, the* and *meeting* are the context feature of *attended* (in each case, there are also source-side context features).

3.2 Bag-of-Words Model

For each side of a translation rule pair, its context words are all collected from the training data, and two "bags-of-words" which consist of collections of source and target context words co-occurring with the rule's source and target sides are created.

$$B_f = \{f_1, f_2, ..., f_I\}$$
$$B_e = \{e_1, e_2, ..., e_J\} \quad (1)$$

where $f_i (1 \le i \le I)$ are source context words which co-occur with the source side of rule α, and $e_j (1 \le j \le J)$ are target context words which co-occur with the target side of rule γ.

Therefore, we can represent source and target sides of the rule by vectors $\vec{v}_f$ and $\vec{v}_e$ as in Equation (2):

$$\vec{v}_f = \{w_{f_1}, w_{f_2}, ..., w_{f_I}\}$$
$$\vec{v}_e = \{w_{e_1}, w_{e_2}, ..., w_{e_J}\} \quad (2)$$

where w_{f_i} and w_{e_j} are values for each source and target context feature; normally, these values are based on the counts of the words in the corresponding bags.

3.3 Feature Weighting Schemes

We use pointwise mutual information (Church et al., 1990) to compute the feature values. Let c ($c \in B_f$ or $c \in B_e$) be a context word and $F(r,c)$ be the frequency count of a rule r (α or γ) co-occurring with the context word c. The pointwise mutual information $MI(r,c)$ is defined as:

$$w(r,c) = MI(r,c) = \frac{\log \frac{F(r,c)}{N}}{\log \frac{F(r)}{N} \times \log \frac{F(c)}{N}} \quad (3)$$

where N is the total frequency counts of all rules and their context words. Since we are using this value as a weight, following (Turney, 2001), we drop *log, N* and $F(r)$. Thus (3) simplifies to:

$$w(r,c) = \frac{F(r,c)}{F(c)} \quad (4)$$

It can be seen as an estimate of $P(r \mid c)$, the empirical probability of observing r given c.

A problem with $P(r \mid c)$ is that it is biased towards infrequent words/features. We therefore smooth $w(r,c)$ with *add-k* smoothing:

$$w(r,c) = \frac{F(r,c)+k}{\sum_{i=1}^{R}(F(r_i,c)+k)} = \frac{F(r,c)+k}{F(c)+kR} \quad (5)$$

where k is a tunable global smoothing constant, and R is the number of rules.

4 Similarity Functions

There are many possibilities for calculating similarities between bags-of-words in different languages. We consider IBM model 1 probabilities and cosine distance similarity functions.

4.1 IBM Model 1 Probabilities

For the IBM model 1 similarity function, we take the geometric mean of symmetrized conditional IBM model 1 (Brown et al., 1993) bag probabilities, as in Equation (6).

$$sim(\alpha, \gamma) = sqrt(P(B_f \mid B_e) \cdot P(B_e \mid B_f)) \quad (6)$$

To compute $P(B_f \mid B_e)$, IBM model 1 assumes that all source words are conditionally independent, so that:

$$P(B_f \mid B_e) = \prod_{i=1}^{I} p(f_i \mid B_e) \quad (7)$$

To compute, we use a "Noisy-OR" combination which has shown better performance than standard IBM model 1 probability, as described in (Zens and Ney, 2004):

$$p(f_i \mid B_e) = 1 - p(\bar{f}_i \mid B_e) \quad (8)$$

$$p(f_i \mid B_e) \approx 1 - \prod_{j=1}^{J}(1 - p(f_i \mid e_j)) \quad (9)$$

where $p(\bar{f}_i \mid B_e)$ is the probability that f_i is *not* in the translation of B_e, and is the IBM model 1 probability.

4.2 Vector Space Mapping

A common way to calculate semantic similarity is by vector space cosine distance; we will also

use this similarity function in our algorithm. However, the two vectors in Equation (2) cannot be directly compared because the axes of their spaces represent different words in different languages, and also their dimensions *I* and *J* are not assured to be the same. Therefore, we need to first map a vector into the space of the other vector, so that the similarity can be calculated. Fung (1998) and Rapp (1999) map the vector one-dimension-to-one-dimension (a context word is a dimension in each vector space) from one language to another language via an initial bilingual dictionary. We follow (Zhao et al., 2004) to do vector space mapping.

Our goal is – given a source pattern – to distinguish between the senses of its associated target patterns. Therefore, we map all vectors in target language into the vector space in the source language. What we want is a representation $\vec{v}_a$ in the source language space of the target vector $\vec{v}_e$. To get $\vec{v}_a$, we can let $w_a^{f_i}$, the weight of the i^{th} source feature, be a linear combination over target features. That is to say, given a source feature weight for f_i, each target feature weight is linked to it with some probability. So that we can calculate a transformed vector from the target vectors by calculating weights $w_a^{f_i}$ using a translation lexicon:

$$w_a^{f_i} = \sum_{j=1}^{J} \Pr(f_i \mid e_j) w_{e_j} \tag{10}$$

where $p(f_i \mid e_j)$ is a lexical probability (we use IBM model 1 probability). Now the source vector and the mapped vector $\vec{v}_a$ have the same dimensions as shown in (11):

$$\begin{aligned} \vec{v}_f &= \{w_{f_1}, w_{f_2}, ..., w_{f_I}\} \\ \vec{v}_a &= \{w_a^{f_1}, w_a^{f_2}, ..., w_a^{f_I}\} \end{aligned} \tag{11}$$

4.3 Naïve Cosine Distance Similarity

The standard cosine distance is defined as the inner product of the two vectors $\vec{v}_f$ and $\vec{v}_a$ normalized by their norms. Based on Equation (10) and (11), it is easy to derive the similarity as follows:

$$\begin{aligned} sim(\alpha,\gamma) &= \cos(\vec{v}_f, \vec{v}_a) = \frac{\vec{v}_f \cdot \vec{v}_a}{|\vec{v}_f| \cdot |\vec{v}_a|} \\ &= \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} w_{f_i} \Pr(f_i \mid e_j) w_{e_j}}{sqrt(\sum_{I=1}^{I} w_{f_i}^2) sqrt(\sum_{i=1}^{I} {w_a^{f_i}}^2)} \end{aligned} \tag{12}$$

where *I* and *J* are the number of the words in source and target bag-of-words; w_{f_i} and w_{e_j} are values of source and target features; $w_a^{f_i}$ is the transformed weight mapped from all target features to the source dimension at word f_i.

4.4 Improved Similarity Function

To incorporate more information than the original similarity functions – IBM model 1 probabilities in Equation (6) and naïve cosine distance similarity function in Equation (12) – we refine the similarity function and propose a new algorithm.

As shown in Figure 2, suppose that we have a rule pair (α, γ). C_f^{full} and C_e^{full} are the contexts extracted according to the definition in section 3 from the full training data for α and for γ, respectively. C_f^{cooc} and C_e^{cooc} are the contexts for α and γ when α and γ co-occur. Obviously, they satisfy the constraints: $C_f^{cooc} \subseteq C_f^{full}$ and $C_e^{cooc} \subseteq C_e^{full}$. Therefore, the original similarity functions are to compare the two context vectors built on full training data directly, as shown in Equation (13).

$$sim(\alpha,\gamma) = sim(C_f^{full}, C_e^{full}) \tag{13}$$

Then, we propose a new similarity function as follows:

$$\begin{aligned} &sim(\alpha,\gamma) = \\ &sim(C_f^{full}, C_f^{cooc})^{\lambda_1} \cdot sim(C_f^{cooc}, C_e^{cooc})^{\lambda_2} \cdot sim(C_e^{full}, C_e^{cooc})^{\lambda_3} \end{aligned} \tag{14}$$

where the parameters λ_i (i=1,2,3) can be tuned via minimal error rate training (MERT) (Och, 2003).

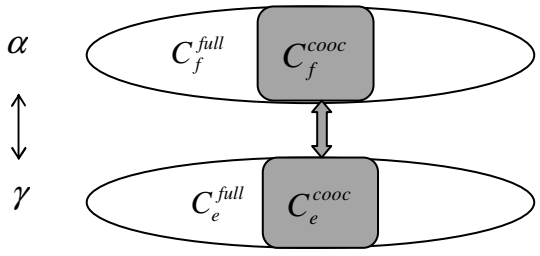


Figure 2: contexts for rule α and γ.

A unit's sense is defined by all its contexts in the whole training data; it may have a lot of different senses in the whole training data. However, when it is linked with another unit in the other language, its sense pool is constrained and is just

a subset of the whole sense set. $sim(C_f^{full}, C_f^{cooc})$ is the metric which evaluates the similarity between the whole sense pool of α and the sense pool when α co-occurs with γ; $sim(C_e^{full}, C_e^{cooc})$ is the analogous similarity metric for γ. They range from 0 to 1. These two metrics both evaluate the similarity for two vectors in the same language, so using cosine distance to compute the similarity is straightforward. And we can set a relatively large size for the vector, since it is not necessary to do vector mapping as the vectors are in the same language. $sim(C_f^{cooc}, C_e^{cooc})$ computes the similarity between the context vectors when α and γ co-occur. We may compute $sim(C_f^{cooc}, C_e^{cooc})$ by using IBM model 1 probability and cosine distance similarity functions as Equation (6) and (12). Therefore, on top of the degree of bilingual semantic similarity between a source and a target translation unit, we have also incorporated the monolingual semantic similarity between all occurrences of a source or target unit, and that unit's occurrence as part of the given rule, into the sense similarity measure.

5 Experiments

We evaluate the algorithm of bilingual sense similarity via machine translation. The sense similarity scores are used as feature functions in the translation model.

5.1 Data

We evaluated with different language pairs: Chinese-to-English, and German-to-English. For Chinese-to-English tasks, we carried out the experiments in two data conditions. The first one is the *large data* condition, based on training data for the NIST [2] 2009 evaluation Chinese-to-English track. In particular, all the allowed bilingual corpora except the *UN corpus* and *Hong Kong Hansard corpus* have been used for estimating the translation model. The second one is the *small data* condition where only the *FBIS*[3] corpus is used to train the translation model. We trained two language models: the first one is a 4-gram LM which is estimated on the target side of the texts used in the *large data* condition. The second LM is a 5-gram LM trained on the so-called English *Gigaword corpus*. Both language models are used for both tasks.

We carried out experiments for translating Chinese to English. We use the same development and test sets for the two data conditions. We first created a development set which used mainly data from the NIST 2005 test set, and also some balanced-genre web-text from the NIST training material. Evaluation was performed on the NIST 2006 and 2008 test sets. Table 1 gives figures for training, development and test corpora; |S| is the number of the sentences, and |W| is the number of running words. Four references are provided for all dev and test sets.

			Chi	Eng
Parallel Train	Large Data	|S|	3,322K	
		|W|	64.2M	62.6M
	Small Data	|S|	245K	
		|W|	9.0M	10.5M
Dev		|S|	1,506	1,506×4
Test	NIST06	|S|	1,664	1,664×4
	NIST08	|S|	1,357	1,357×4
Gigaword		|S|	-	11.7M

Table 1: Statistics of training, dev, and test sets for Chinese-to-English task.

For German-to-English tasks, we used WMT 2006[4] data sets. The parallel training data contains 21 million target words; both the dev set and test set contain 2000 sentences; one reference is provided for each source input sentence. Only the target-language half of the parallel training data are used to train the language model in this task.

5.2 Results

For the baseline, we train the translation model by following (Chiang, 2005; Chiang, 2007) and our decoder is *Joshua*[5], an open-source hierarchical phrase-based machine translation system written in Java. Our evaluation metric is IBM BLEU (Papineni et al., 2002), which performs case-insensitive matching of *n*-grams up to $n = 4$. Following (Koehn, 2004), we use the bootstrap-resampling test to do significance testing.

By observing the results on dev set in the additional experiments, we first set the smoothing constant k in Equation (5) to 0.5.

Then, we need to set the sizes of the vectors to balance the computing time and translation accu-

[2] http://www.nist.gov/speech/tests/mt
[3] LDC2003E14
[4] http://www.statmt.org/wmt06/
[5] http://www.cs.jhu.edu/~ccb/joshua/index.html

racy, *i.e.*, we keep only the top N context words with the highest feature value for each side of a rule [6]. In the following, we use "Alg1" to represent the original similarity functions which compare the two context vectors built on full training data, as in Equation (13); while we use "Alg2" to represent the improved similarity as in Equation (14). "IBM" represents IBM model 1 probabilities, and "COS" represents cosine distance similarity function.

After carrying out a series of additional experiments on the *small data* condition and observing the results on the dev set, we set the size of the vector to 500 for Alg1; while for Alg2, we set the sizes of C_f^{full} and C_e^{full} N_1 to 1000, and the sizes of C_f^{cooc} and C_e^{cooc} N_2 to 100.

The sizes of the vectors in Alg2 are set in the following process: first, we set N_2 to 500 and let N_1 range from 500 to 3,000, we observed that the dev set got best performance when N_1 was 1000; then we set N_1 to 1000 and let N_1 range from 50 to 1000, we got best performance when N_1=100. We use this setting as the default setting in all remaining experiments.

Algorithm	NIST'06	NIST'08
Baseline	27.4	21.2
Alg1 IBM	27.8*	21.5
Alg1 COS	27.8*	21.5
Alg2 IBM	27.9*	21.6*
Alg2 COS	**28.1****	**21.7***

Table 2: Results (BLEU%) of *small data* Chinese-to-English NIST task. Alg1 represents the original similarity functions as in Equation (13); while Alg2 represents the improved similarity as in Equation (14). IBM represents IBM model 1 probability, and COS represents cosine distance similarity function. * or ** means result is significantly better than the baseline ($p < 0.05$ or $p < 0.01$, respectively).

	Ch-En		De-En
Algorithm	NIST'06	NIST'08	Test'06
Baseline	31.0	23.8	26.9
Alg2 IBM	31.5*	24.5**	27.2*
Alg2 COS	**31.6****	**24.5****	**27.3***

Table 3: Results (BLEU%) of *large data* Chinese-to-English NIST task and German-to-English WMT task.

[6] We have also conducted additional experiments by removing the stop words from the context vectors; however, we did not observe any consistent improvement. So we filter the context vectors by only considering the feature values.

Table 2 compares the performance of Alg1 and Alg2 on the Chinese-to-English *small data* condition. Both Alg1 and Alg2 improved the performance over the baseline, and Alg2 obtained slight and consistent improvements over Alg1. The improved similarity function Alg2 makes it possible to incorporate monolingual semantic similarity on top of the bilingual semantic similarity, thus it may improve the accuracy of the similarity estimate. Alg2 significantly improved the performance over the baseline. The Alg2 cosine similarity function got 0.7 BLEU-score (p<0.01) improvement over the baseline for NIST 2006 test set, and a 0.5 BLEU-score (p<0.05) for NIST 2008 test set.

Table 3 reports the performance of Alg2 on Chinese-to-English NIST *large data* condition and German-to-English WMT task. We can see that IBM model 1 and cosine distance similarity function both obtained significant improvement on all test sets of the two tasks. The two similarity functions obtained comparable results.

6 Analysis and Discussion

6.1 Effect of Single Features

In Alg2, the similarity score consists of three parts as in Equation (14): $sim(C_f^{full}, C_f^{cooc})$, $sim(C_e^{full}, C_e^{cooc})$, and $sim(C_f^{cooc}, C_e^{cooc})$; where $sim(C_f^{cooc}, C_e^{cooc})$ could be computed by IBM model 1 probabilities $sim_{IBM}(C_f^{cooc}, C_e^{cooc})$ or cosine distance similarity function $sim_{COS}(C_f^{cooc}, C_e^{cooc})$. Therefore, our first study is to determine which one of the above four features has the most impact on the result. Table 4 shows the results obtained by using each of the 4 features. First, we can see that $sim_{IBM}(C_f^{cooc}, C_e^{cooc})$ always gives a better improvement than $sim_{COS}(C_f^{cooc}, C_e^{cooc})$. This is because $sim_{IBM}(C_f^{cooc}, C_e^{cooc})$ scores are more diverse than the latter when the number of context features is small (there are many rules that have only a few contexts.) For an extreme example, suppose that there is only one context word in each vector of source and target context features, and the translation probability of the two context words is not 0. In this case, $sim_{IBM}(C_f^{cooc}, C_e^{cooc})$ reflects the translation probability of the context word pair, while $sim_{COS}(C_f^{cooc}, C_e^{cooc})$ is always 1.

Second, $sim(C_f^{full}, C_f^{cooc})$ and $sim(C_e^{full}, C_e^{cooc})$ also give some improvements even when used

independently. For a possible explanation, consider the following example. The Chinese word "红" can translate to "red", "communist", or "hong" (the transliteration of 红, when it is used in a person's name). Since these translations are likely to be associated with very different source contexts, each will have a low $sim(C_f^{full}, C_f^{cooc})$ score. Another Chinese word 小溪 may translate into synonymous words, such as "brook", "stream", and "rivulet", each of which will have a high $sim(C_f^{full}, C_f^{cooc})$ score. Clearly, 红 is a more "dangerous" word than 小溪, since choosing the wrong translation for it would be a bad mistake. But if the two words have similar translation distributions, the system cannot distinguish between them. The monolingual similarity scores give it the ability to avoid "dangerous" words, and choose alternatives (such as larger phrase translations) when available.

Third, the similarity function of Alg2 consistently achieved further improvement by incorporating the monolingual similarities computed for the source and target side. This confirms the effectiveness of our algorithm.

	CE_LD		CE_SD	
testset (NIST)	'06	'08	'06	'08
Baseline	31.0	23.8	27.4	21.2
$sim(C_f^{full}, C_f^{cooc})$	31.1	24.3	27.5	21.3
$sim(C_e^{full}, C_e^{cooc})$	31.1	23.9	27.9	21.5
$sim_{IBM}(C_f^{cooc}, C_e^{cooc})$	31.4	24.3	27.9	21.5
$sim_{COS}(C_f^{cooc}, C_e^{cooc})$	31.2	23.9	27.7	21.4
Alg2 IBM	31.5	24.5	27.9	21.6
Alg2 COS	**31.6**	**24.5**	**28.1**	**21.7**

Table 4: Results (BLEU%) of Chinese-to-English *large data* (CE_LD) and *small data* (CE_SD) NIST task by applying one feature.

6.2 Effect of Combining the Two Similarities

We then combine the two similarity scores by using both of them as features to see if we could obtain further improvement. In practice, we use the four features in Table 4 together.

Table 5 reports the results on the *small data* condition. We observed further improvement on dev set, but failed to get the same improvements on test sets or even lost performance. Since the IBM+COS configuration has one extra feature, it is possible that it overfits the dev set.

Algorithm	Dev	NIST'06	NIST'08
Baseline	20.2	27.4	21.2
Alg2 IBM	20.5	27.9	21.6
Alg2 COS	20.6	28.1	21.7
Alg2 IBM+COS	20.8	27.9	21.5

Table 5: Results (BLEU%) for combination of two similarity scores. Further improvement was only obtained on dev set but not on test sets.

6.3 Comparison with Simple Contextual Features

Now, we try to answer the question: can the similarity features computed by the function in Equation (14) be replaced with some other simple features? We did additional experiments on *small data* Chinese-to-English task to test the following features: (15) and (16) represent the sum of the counts of the context words in C^{full}, while (17) represents the proportion of words in the context of α that appeared in the context of the rule (α, γ); similarly, (18) is related to the properties of the words in the context of γ.

$$N_f(\alpha) = \sum_{f_i \in C_f^{full}} F(\alpha, f_i) \quad (15)$$

$$N_e(\gamma) = \sum_{e_j \in C_e^{full}} F(\gamma, e_j) \quad (16)$$

$$E_f(\alpha, \gamma) = \frac{\sum_{f_i \in C_f^{cooc}} F(\alpha, f_i)}{N_f(\alpha)} \quad (17)$$

$$E_e(\alpha, \gamma) = \frac{\sum_{e_j \in C_e^{cooc}} F(\gamma, e_j)}{N_e(\gamma)} \quad (18)$$

where $F(\alpha, f_i)$ and $F(\gamma, e_j)$ are the frequency counts of rule α or γ co-occurring with the context word f_i or e_j respectively.

Feature	Dev	NIST'06	NIST'08
Baseline	20.2	27.4	21.2
$+N_f$	20.5	27.6	21.4
$+N_e$	20.5	27.5	21.3
$+E_f$	20.4	27.5	21.2
$+E_e$	20.4	27.3	21.2
$+N_f+N_e$	20.5	27.5	21.3

Table 6: Results (BLEU%) of using simple features based on context on *small data* NIST task. Some improvements are obtained on dev set, but there was no significant effect on the test sets.

Table 6 shows results obtained by adding the above features to the system for the *small data*

condition. Although all these features have obtained some improvements on dev set, there was no significant effect on the test sets. This means simple features based on context, such as the sum of the counts of the context features, are not as helpful as the sense similarity computed by Equation (14).

6.4 Null Context Feature

There are two cases where no context word can be extracted according to the definition of context in Section 3.1. The first case is when a rule pair is always a full sentence-pair in the training data. The second case is when for some rule pairs, either their source or target contexts are out of the span limit of the initial phrase, so that we cannot extract contexts for those rule-pairs. For Chinese-to-English NIST task, there are about 1% of the rules that do not have contexts; for German-to-English task, this number is about 0.4%. We assign a uniform number as their bilingual sense similarity score, and this number is tuned through MERT. We call it the *null context* feature. It is included in all the results reported from Table 2 to Table 6. In Table 7, we show the weight of the *null context* feature tuned by running MERT in the experiments reported in Section 5.2. We can learn that penalties always discourage using those rules which have no context to be extracted.

	Task		
Alg.	CE_SD	CE_LD	DE
Alg2 IBM	-0.09	-0.37	-0.15
Alg2 COS	-0.59	-0.42	-0.36

Table 7: Weight learned for employing the *null context* feature. CE_SD, CE_LD and DE are Chinese-to-English *small data* task, *large data* task and German-to-English task respectively.

6.5 Discussion

Our aim in this paper is to characterize the semantic similarity of bilingual hierarchical rules. We can make several observations concerning our features:

1) Rules that are largely syntactic in nature, such as *的 X ||| the X of*, will have very diffuse "meanings" and therefore lower similarity scores. It could be that the gains we obtained come simply from biasing the system against such rules. However, the results in table 6 show that this is unlikely to be the case: features that just count context words help very little.

2) In addition to bilingual similarity, Alg2 relies on the degree of monolingual similarity between the sense of a source or target unit within a rule, and the sense of the unit in general. This has a bias in favor of less ambiguous rules, i.e. rules involving only units with closely related meanings. Although this bias is helpful on its own, possibly due to the mechanism we outline in section 6.1, it appears to have a synergistic effect when used along with the bilingual similarity feature.

3) Finally, we note that many of the features we use for capturing similarity, such as the context "*the, of*" for instantiations of *X* in the unit *the X of*, are arguably more syntactic than semantic. Thus, like other "semantic" approaches, ours can be seen as blending syntactic and semantic information.

7 Related Work

There has been extensive work on incorporating semantics into SMT. Key papers by Carpuat and Wu (2007) and Chan et al (2007) showed that word-sense disambiguation (WSD) techniques relying on source-language context can be effective in selecting translations in phrase-based and hierarchical SMT. More recent work has aimed at incorporating richer disambiguating features into the SMT log-linear model (Gimpel and Smith, 2008; Chiang et al, 2009); predicting coherent sets of target words rather than individual phrase translations (Bangalore et al, 2009; Mauser et al, 2009); and selecting applicable rules in hierarchical (He et al, 2008) and syntactic (Liu et al, 2008) translation, relying on source as well as target context. Work by Wu and Fung (2009) breaks new ground in attempting to match semantic roles derived from a semantic parser across source and target languages.

Our work is different from all the above approaches in that we attempt to discriminate among hierarchical rules based on: 1) the degree of bilingual semantic similarity between source and target translation units; and 2) the monolingual semantic similarity between occurrences of source or target units as part of the given rule, and in general. In another words, WSD explicitly tries to choose a translation given the current source context, while our work rates rule pairs independent of the current context.

8 Conclusions and Future Work

In this paper, we have proposed an approach that uses the vector space model to compute the sense

similarity for terms from parallel corpora and applied it to statistical machine translation. We saw that the bilingual sense similarity computed by our algorithm led to significant improvements. Therefore, we can answer the questions proposed in Section 1. We have shown that the sense similarity computed between units from parallel corpora by means of our algorithm is helpful for at least one multilingual application: statistical machine translation.

Finally, although we described and evaluated bilingual sense similarity algorithms applied to a hierarchical phrase-based system, this method is also suitable for syntax-based MT systems and phrase-based MT systems. The only difference is the definition of the context. For a syntax-based system, the context of a rule could be defined similarly to the way it was defined in the work described above. For a phrase-based system, the context of a phrase could be defined as its surrounding words in a given size window. In our future work, we may try this algorithm on syntax-based MT systems and phrase-based MT systems with different context features. It would also be possible to use this technique during training of an SMT system – for instance, to improve the bilingual word alignment or reduce the training data noise.

References

S. Bangalore, S. Kanthak, and P. Haffner. 2009. Statistical Machine Translation through Global Lexical Selection and Sentence Reconstruction. In: Goutte et al (ed.), *Learning Machine Translation*. MIT Press.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra & R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2) 263-312.

J. Bullinaria and J. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39 (3), 510–526.

M. Carpuat and D. Wu. 2007. Improving Statistical Machine Translation using Word Sense Disambiguation. In: *Proceedings of EMNLP*, Prague.

M. Carpuat. 2009. One Translation per Discourse. In: *Proceedings of NAACL HLT Workshop on Semantic Evaluations*, Boulder, CO.

Y. Chan, H. Ng and D. Chiang. 2007. Word Sense Disambiguation Improves Statistical Machine Translation. In: *Proceedings of ACL*, Prague.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In: *Proceedings of ACL*, pp. 263–270.

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

D. Chiang, W. Wang and K. Knight. 2009. 11,001 new features for statistical machine translation. In: *Proc. NAACL HLT*, pp. 218–226.

K. W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

W. B. Frakes and R. Baeza-Yates, editors. 1992. *Information Retrieval, Data Structure and Algorithms*. Prentice Hall.

P. Fung. 1998. A statistical view on bilingual lexicon extraction: From parallel corpora to non-parallel corpora. In: *Proceedings of AMTA*, pp. 1–17. Oct. Langhorne, PA, USA.

J. Gimenez and L. Marquez. 2009. Discriminative Phrase Selection for SMT. In: Goutte et al (ed.), *Learning Machine Translation*. MIT Press.

K. Gimpel and N. A. Smith. 2008. Rich Source-Side Context for Statistical Machine Translation. In: *Proceedings of WMT*, Columbus, OH.

Z. Harris. 1954. Distributional structure. *Word*, 10(23): 146-162.

Z. He, Q. Liu, and S. Lin. 2008. Improving Statistical Machine Translation using Lexicalized Rule Selection. In: *Proceedings of COLING*, Manchester, UK.

D. Hindle. 1990. Noun classification from predicate-argument structures. In: *Proceedings of ACL*. pp. 268-275. Pittsburgh, PA.

P. Koehn, F. Och, D. Marcu. 2003. Statistical Phrase-Based Translation. In: *Proceedings of HLT-NAACL*. pp. 127-133, Edmonton, Canada

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In: *Proceedings of EMNLP*, pp. 388–395. July, Barcelona, Spain.

T. Landauer and S. T. Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*. 104:211-240.

Z. Li, C. Callison-Burch, C. Dyer, J. Ganitkevitch, S. Khudanpur, L. Schwartz, W. Thornton, J. Weese and O. Zaidan, 2009. Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In: *Proceedings of the WMT*. March. Athens, Greece.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In: *Proceedings of COLING/ACL-98*. pp. 768-774. Montreal, Canada.

Q. Liu, Z. He, Y. Liu and S. Lin. 2008. Maximum Entropy based Rule Selection Model for Syntax-based Statistical Machine Translation. In: *Proceedings of EMNLP*, Honolulu, Hawaii.

K. Lund, and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28 (2), 203–208.

A. Mauser, S. Hasan and H. Ney. 2009. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In: *Proceedings of EMNLP*, Singapore.

F. Och. 2003. Minimum error rate training in statistical machine translation. In: *Proceedings of ACL*. Sapporo, Japan.

S. Pado and M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33 (2), 161–199.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In: *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 613–619. Edmonton, Canada.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pp. 311–318. July. Philadelphia, PA, USA.

R. Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In: *Proceedings of ACL*, pp. 519–526. June. Maryland.

G. Salton and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

P. Turney. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In: *Proceedings of the Twelfth European Conference on Machine Learning*, pp. 491–502, Berlin, Germany.

D. Wu and P. Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In: *Proceedings of NAACL/HLT*, Boulder, CO.

D. Yuret and M. A. Yatbaz. 2009. The Noisy Channel Model for Unsupervised Word Sense Disambiguation. In: *Computational Linguistics*. Vol. 1(1) 1-18.

R. Zens and H. Ney. 2004. Improvements in phrase-based statistical machine translation. In: *Proceedings of NAACL-HLT*. Boston, MA.

B. Zhao, S. Vogel, M. Eck, and A. Waibel. 2004. Phrase pair rescoring with term weighting for statistical machine translation. In *Proceedings of EMNLP*, pp. 206–213. July. Barcelona, Spain.

Untangling the Cross-Lingual Link Structure of Wikipedia

Gerard de Melo
Max Planck Institute for Informatics
Saarbrücken, Germany
demelo@mpi-inf.mpg.de

Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

Abstract

Wikipedia articles in different languages are connected by interwiki links that are increasingly being recognized as a valuable source of cross-lingual information. Unfortunately, large numbers of links are imprecise or simply wrong. In this paper, techniques to detect such problems are identified. We formalize their removal as an optimization task based on graph repair operations. We then present an algorithm with provable properties that uses linear programming and a region growing technique to tackle this challenge. This allows us to transform Wikipedia into a much more consistent multilingual register of the world's entities and concepts.

1 Introduction

Motivation. The open community-maintained encyclopedia Wikipedia has not only turned the Internet into a more useful and linguistically diverse source of information, but is also increasingly being used in computational applications as a large-scale source of linguistic and encyclopedic knowledge. To allow cross-lingual navigation, Wikipedia offers cross-lingual *interwiki* links that for instance connect the Indonesian article about Albert Einstein to the corresponding articles in over 100 other languages. Such links are extraordinarily valuable for cross-lingual applications.

In the ideal case, a set of articles connected directly or indirectly via such links would all describe the same entity or concept. Due to conceptual drift, different granularities, as well as mistakes made by editors, we frequently find concepts as different as *economics* and *manager* in the same connected component. Filtering out inaccurate links enables us to exploit Wikipedia's multilinguality in a much safer manner and allows us to create a multilingual register of named entities.

Contribution. Our research contributions are: 1) We identify criteria to detect inaccurate connections in Wikipedia's cross-lingual link structure. 2) We formalize the task of removing such links as an optimization problem. 3) We introduce an algorithm that attempts to repair the cross-lingual graph in a minimally invasive way. This algorithm has an approximation guarantee with respect to optimal solutions. 4) We show how this algorithm can be used to combine all editions of Wikipedia into a single large-scale multilingual register of named entities and concepts.

2 Detecting Inaccurate Links

In this paper, we model the union of cross-lingual links provided by all editions of Wikipedia as an undirected graph $G = (V, E)$ with edge weights $w(e)$ for $e \in E$. In our experiments, we simply honour each individual link equally by defining $w(e) = 2$ if there are reciprocal links between the two pages, 1 if there is a single link, and 0 otherwise. However, our framework is flexible enough to deal with more advanced weighting schemes, e.g. one could easily plug in cross-lingual measures of semantic relatedness between article texts.

It turns out that an astonishing number of connected components in this graph harbour inaccurate links between articles. For instance, the Esperanto article '*Germana Imperiestro*' is about German emporers and another Esperanto article '*Germana Imperiestra Regno*' is about the German Empire, but, as of June 2010, both are linked to the English and German articles about the German Empire. Over time, some inaccurate links may be fixed, but in this and in large numbers of other cases, the imprecise connection has persisted for many years. In order to detect such cases, we need to have some way of specifying that two articles are likely to be distinct.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 844–853,
Uppsala, Sweden, 11-16 July 2010.

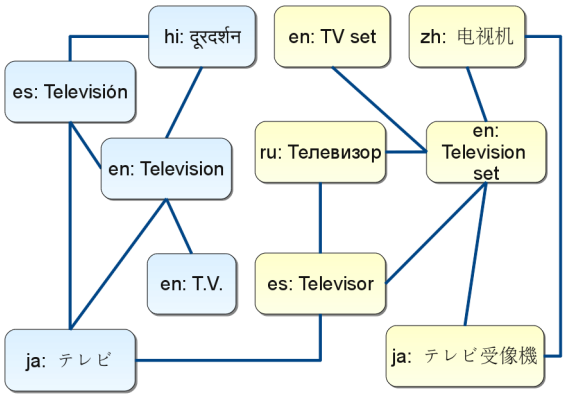


Figure 1: Connected component with inaccurate links (simplified)

2.1 Distinctness Assertions

Figure 1 shows a connected component that conflates the concept of television as a medium with the concept of TV sets as devices. Among other things, we would like to state that '*Television*' and '*T.V.*' are distinct from '*Television set*' and '*TV set*'. In general, we may have several sets of entities $D_{i,1}, \ldots, D_{i,l_i}$, for which we assume that any two entities u,v from different sets are pairwise distinct with some degree of confidence or weight. In our example, $D_{i,1} = \{$'*Television*','*T.V.*'$\}$ would be one set, and $D_{i,2} = \{$'*Television set*','*TV set*'$\}$ would be another set, which means that we are assuming '*Television*', for example, to be distinct from both '*Television set*' and '*TV set*'.

Definition 1. *(Distinctness Assertions) Given a set of nodes V, a* distinctness assertion *is a collection $D_i = (D_{i,1}, \ldots, D_{i,l_i})$ of pairwise disjoint (i.e. $D_{i,j} \cap D_{i,k} = \emptyset$ for $j \neq k$) subsets $D_{i,j} \subset V$ that expresses that any two nodes $u \in D_{i,j}$, $v \in D_{i,k}$ from different subsets ($j \neq k$) are asserted to be distinct from each other with some weight $w(D_i) \in \mathbb{R}$.*

We found that many components with inaccurate links can be identified automatically with the following distinctness assertions.

Criterion 1. *(Distinctness between articles from the same Wikipedia edition)* For each language-specific edition of Wikipedia, a separate assertion $(D_{i,1}, D_{i,2}, \ldots)$ can be made, where each $D_{i,j}$ contains an individual *article* together with its respective redirection pages. Two articles from the same Wikipedia very likely describe distinct concepts unless they are redirects of each other. For example, '*Georgia (country)*' is distinct from '*Georgia (U.S. State)*'. Additionally, there are also redirects that are clearly marked by a category or template as involving topic drift, e.g. redirects from songs to albums or artists, from products to companies, etc. We keep such redirects in a $D_{i,j}$ distinct from the one of their redirect targets.

Criterion 2. *(Distinctness between categories from the same Wikipedia edition)* For each language-specific edition of Wikipedia, a separate assertion $(D_{i,1}, D_{i,2}, \ldots)$ is made, where each $D_{i,j}$ contains a *category* page together with any redirects. For instance, '*Category:Writers*' is distinct from '*Category:Writing*'.

Criterion 3. *(Distinctness for links with anchor identifiers)* The English '*Division by zero*', for instance, links to the German '*Null#Division*'. The latter is only a part of a larger article about the number zero in general, so we can make a distinctness assertion to separate '*Division by zero*' from '*Null*'. In general, for each interwiki link or redirection with an anchor identifier, we add an assertion $(D_{i,1}, D_{i,2})$ where $D_{i,1}$,$D_{i,2}$ represent the respective articles without anchor identifiers.

These three types of distinctness assertions are instantiated for all articles and categories of all Wikipedia editions. The assertion weights are tunable; the simplest choice is using a uniform weight for all assertions (note that these weights are different from the edge weights in the graph). We will revisit this issue in our experiments.

2.2 Enforcing Consistency

Given a graph G representing cross-lingual links between Wikipedia pages, as well as distinctness assertions $D_1, \ldots, D_n$ with weights $w(D_i)$, we may find that nodes that are asserted to be distinct are in the same connected component. We can then try to apply repair operations to reconcile the graph's link structure with the distinctness asssertions and obtain global consistency. There are two ways to modify the input, and for each we can also consider the corresponding weights as a sort of *cost* that quantifies how much we are changing the original input:

a) **Edge cutting:** We may remove an edge $e \in E$ from the graph, paying cost $w(e)$.
b) **Distinctness assertion relaxation:** We may remove a node $v \in V$ from a distinctness assertion D_i, paying cost $w(D_i)$.

Removing edges allows us to split connected components into multiple smaller components, thereby ensuring that two nodes asserted to be distinct are no longer connected directly or indirectly. In Figure 1, for instance, we could delete the edge from the Spanish '*TV set*' article to the Japanese '*television*' article. In constrast, removing nodes from distinctness assertions means that we decide to give up our claim of them being distinct, instead allowing them to share a connected component.

Our reliance on costs is based on the assumption that the link structure or topology of the graph provides the best indication of which cross-lingual links to remove. In Figure 1, we have distinctness assertions between nodes in two densely connected clusters that are tied together only by a single spurious link. In such cases, edge removals can easily yield separate connected components. When, however, the two nodes are strongly connected via many different paths with high weights, we may instead opt for removing one of the two nodes from the distinctness assertion.

The aim will be to balance the costs for removing edges from the graph with the costs for removing nodes from distinctness assertions to produce a consistent solution with a minimal total repair cost. We accommodate our knowledge about distinctness while staying as close as possible to what Wikipedia provides as input.

This can be formalized as the **Weighted Distinctness-Based Graph Separation (WDGS)** problem. Let G be an undirected graph with a set of vertices V and a set of edges E weighted by $w : E \to \mathbb{R}$. If we use a set $C \subseteq V$ to specify which edges we want to cut from the original graph, and sets U_i to specify which nodes we want to remove from distinctness assertions, we can begin by defining WDGS solutions as follows.

Definition 2. *(**WDGS** Solution). Given a graph $G = (V, E)$ and n distinctness assertions $D_1, \ldots, D_n$, a tuple $(C, U_1, \ldots, U_n)$ is a valid* WDGS *solution if and only if $\forall i, j, k \neq j, u \in D_{i,j} \setminus U_i$, $v \in D_{i,k} \setminus U_i$: $\mathrm{P}(u, v, E \setminus C) = \emptyset$, i.e. the set of paths from u to v in the graph $(V, E \setminus C)$ is empty.*

Definition 3. *(**WDGS** Cost). Let $w : E \to \mathbb{R}$ be a weight function for edges $e \in E$, and $w(D_i)$ $(i = 1 \ldots n)$ be weights for the distinctness assertions. The (total) cost of a* WDGS *solution $S = (C, U_1, \ldots, U_n)$ is then defined as*

$$c(S) = c(C, U_1, \ldots, U_n) = \left[\sum_{e \in C} w(e)\right] + \left[\sum_{i=1}^{n} |U_i|\, w(D_i)\right]$$

Definition 4. *(**WDGS**). A* WDGS *problem instance P consists of a graph $G = (V, E)$ with edge weights $w(e)$ and n distinctness assertions D_1, $\ldots$, D_n with weights $w(D_i)$. The objective consists in finding a solution $(C, U_1, \ldots, U_n)$ with minimal cost $c(C, U_1, \ldots, U_n)$.*

It turns out that finding optimal solutions efficiently is a hard problem (proofs in Appendix A).

Theorem 1. WDGS *is NP-hard and APX-hard. If the Unique Games Conjecture (Khot, 2002) holds, then it is NP-hard to approximate* WDGS *within any constant factor $\alpha > 0$.*

3 Approximation Algorithm

Due to the hardness of WDGS, we devise a polynomial-time approximation algorithm with an approximation factor of $4 \ln(nq + 1)$ where n is the number of distinctness assertions and $q = \max_{i,j} |D_{i,j}|$. This means that for all problem instances P, we can guarantee

$$\frac{c(S(P))}{c(S^*(P))} \leq 4 \ln(nq + 1),$$

where $S(P)$ is the solution determined by our algorithm, and $S^*(P)$ is an optimal solution. Note that this approximation guarantee is independent of how long each D_i is, and that it merely represents an upper bound on the worst case scenario. In practice, the results tend to be much closer to the optimum, as will be shown in Section 4.

Our algorithm first solves a linear program (LP) relaxation of the original problem, which gives us hints as to which edges should most likely be cut and which nodes should most likely be removed from distinctness assertions. Note that this is a continuous LP, not an integer linear program (ILP); the latter would not be tractable due to the large number of variables and constraints of the problem. After solving the linear program, a new – extended – graph is constructed and the optimal LP solution is used to define a distance metric on it. The final solution is obtained by smartly selecting regions in this extended graph as the individual output components, employing a region

growing technique in the spirit of the seminal work by Leighton and Rao (1999). Edges that cross the boundaries of these regions are cut.

Definition 5. *Given a* WDGS *instance, we define a linear program of the following form:*

minimize

$$\sum_{e \in E} d_e w(e) + \sum_{i=1}^{n} \sum_{j=1}^{l_i} \sum_{v \in D_{i,j}} u_{i,v} w(D_i)$$

subject to

$$p_{i,j,v} = u_{i,v} \qquad \forall i,\, j{<}l_i,\, v \in D_{i,j} \qquad (1)$$

$$p_{i,j,v} + u_{i,v} \geq 1 \qquad \forall i,\, j{<}l_i,\, v \in \bigcup_{k>j} D_{i,k} \qquad (2)$$

$$p_{i,j,v} \leq p_{i,j,u} + d_e \qquad \forall i,\, j{<}l_i,\, e{=}(u,v) \in E \qquad (3)$$

$$d_e \geq 0 \qquad \forall e \in E \qquad (4)$$

$$u_{i,v} \geq 0 \qquad \forall i,\, v \in \bigcup_{j=1}^{l_i} D_{i,j} \qquad (5)$$

$$p_{i,j,v} \geq 0 \qquad \forall i,\, j{<}l_i,\, v{\in}V \qquad (6)$$

The LP uses decision variables d_e and $u_{i,v}$, and auxiliary variables $p_{i,j,v}$ that we refer to as *potential variables*. The d_e variables indicate whether (in the continuous LP: to what degree) an edge e should be deleted, and the $u_{i,v}$ variables indicate whether (to what degree) v should be removed from a distinctness assertion D_i. The LP objective function corresponds to Definition 3, aiming to minimize the total costs. A potential variable $p_{i,j,v}$ reflects a sort of potential difference between an assertion $D_{i,j}$ and a node v. If $p_{i,j,v} = 0$, then v is still connected to nodes in $D_{i,j}$. Constraints (1) and (2) enforce potential differences between $D_{i,j}$ and all nodes in $D_{i,k}$ with $k > j$. For instance, for distinctness between '*New York City*' and '*New York*' (the state), they might require '*New York*' to have a potential of 1, while '*New York City*' has a potential of 0. The potential variables are tied to the deletion variables d_e for edges in Constraint (3) as well as to the $u_{i,v}$ in Constraints (1) and (2). This means that the potential difference $p_{i,j,v} + u_{i,v} \geq 1$ can only be obtained if edges are deleted on every path between '*New York City*' and '*New York*', or if at least one of these two nodes is removed from the distinctness assertion (by setting the corresponding $u_{i,v}$ to non-zero values). Constraints (4), (5), (6) ensure non-negativity.

Having solved the linear program, the next major step is to convert the optimal LP solution into the final – discrete – solution. We cannot rely on standard rounding methods to turn the optimal fractional values of the d_e and $u_{i,v}$ variables into a valid solution. Often, all solution variables have small values and rounding will merely produce an empty $(C, U_1, \ldots, U_n) = (\emptyset, \emptyset, \ldots, \emptyset)$. Instead, a more sophisticated technique is necessary. The optimal solution of the LP can be used to define an extended graph G' with a distance metric d between nodes. The algorithm then operates on this graph, in each iteration selecting regions that become output components and removing them from the graph. A simple example is shown in Figure 2. The extended graph contains additional nodes and edges representing distinctness assertions. Cutting one of these additional edges corresponds to removing a node from a distinctness assertion.

Definition 6. *Given $G = (V, E)$ and distinctness assertions $D_1, \ldots, D_n$ with weights $w(D_i)$, we define an undirected graph $G' = (V', E')$ where $V' = V \cup \{\mathrm{v}_{i,v} \mid i = 1 \ldots n, w(D_i) > 0, v \in \bigcup_j D_{i,j}\}$, $E' = \{e \in E \mid w(e) > 0\} \cup \{(v, \mathrm{v}_{i,v}) \mid v \in D_{i,j}, w(D_i) > 0\}$. We accordingly extend the definition of $w(e)$ to additionally cover the new edges by defining $w(e) = w(D_i)$ for $e = (v, \mathrm{v}_{i,v})$. We also extend it for sets S of edges by defining $w(S) = \sum_{e \in S} w(e)$. Finally, we define a node distance metric*

$$d(u,v) = \begin{cases} 0 & u = v \\ d_e & (u,v) \in E \\ u_{i,v} & u = \mathrm{v}_{i,v} \\ u_{i,u} & v = \mathrm{v}_{i,u} \\ \min\limits_{p \in \mathrm{P}(u,v,E')} \sum\limits_{(u',v') \in p} d(u',v') & \textit{otherwise,} \end{cases}$$

where $\mathrm{P}(u, v, E')$ denotes the set of acyclic paths between two nodes in E'. We further fix

$$\hat{c}_\mathrm{f} = \sum_{(u,v) \in E'} d(u,v)\, w(e)$$

as the weight of the fractional solution of the LP ($\hat{c}_\mathrm{f}$ is a constant based on the original E', irrespective of later modifications to the graph).

Definition 7. *Around a given node v in G', we consider regions $R(v, r) \subseteq V$ with radius r. The cut $C(v, r)$ of a given region is defined as the set of edges in G' with one endpoint within the region and one outside the region:*

$$R(v,r) = \{v' \in V' \mid d(v,v') \leq r\}$$

$$C(v,r) = \{e \in E' \mid |e \cap R(v,r)| = 1\}$$

For sets of nodes $S \subseteq V$, we define $R(S, r) = \bigcup_{v \in S} R(v, r)$ and $C(S, r) = \bigcup_{v \in S} C(v, r)$.

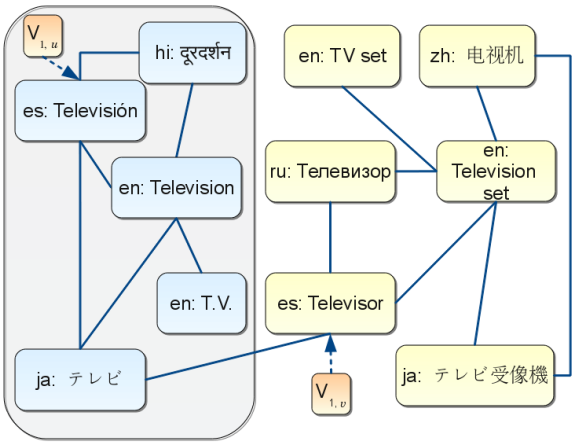


Figure 2: Extended graph with two added nodes $v_{1,u}$, $v_{1,v}$ representing distinctness between '*Televisión*' and '*Televisor*', and a region around $v_{1,u}$ that would cut the link from the Japanese '*Television*' to '*Televisor*'

Definition 8. *Given* $q = \max_{i,j} |D_{i,j}|$*, we approximate the optimal cost of regions as:*

$$\hat{c}(v,r) = \sum_{\substack{e=(u,u')\in E':\\ e\subseteq R(v,r)}} d(u,u')\, w(e) \quad (1)$$
$$+ \sum_{\substack{e\in C(v,r)\\ v'\in e\cap R(v,r)}} (r - d(v,v'))\, w(e)$$

$$\hat{c}(S,r) = \frac{1}{nq}\hat{c}_{\mathrm{f}} + \sum_{v\in S} \hat{c}(v,r) \quad (2)$$

The first summand accounts for the edges entirely within the region, and the second one accounts for the edges in $C(v,r)$ to the extent that they are within the radius. The definition of $\hat{c}(S,r)$ contains an additional slack component that is required for the approximation guarantee proof.

Based on these definitions, Algorithm 3.1 uses the LP solution to construct the extended graph. It then repeatedly, as long as there is an unsatisfied assertion D_i, chooses a set S of nodes containing one node from each relevant $D_{i,j}$. Around the nodes in S it simultaneously grows $|S|$ regions with the same radius, a technique previously suggested by Avidor and Langberg (2007). These regions are essentially output components that determine the solution. Repeatedly choosing the radius that minimizes $\frac{w(C(S,r))}{\hat{c}(S,r)}$ allows us to obtain the approximation guarantee, because the distances in this extended graph are based on the solution of the LP. The properties of this algorithm are given by the following two theorems (proofs in Appendix A).

Theorem 2. *The algorithm yields a* valid WDGS *solution* $(C, U_1, \ldots, U_n)$.

Theorem 3. *The algorithm yields a solution* $(C, U_1, \ldots, U_n)$ *with an approximation factor of* $4\ln(nq+1)$ *with respect to the cost of the optimal* WDGS *solution* $(C^*, U_1^*, \ldots, U_n^*)$*, where* n *is the number of distinctness assertions and* $q = \max_{i,j} |D_{i,j}|$*. This solution can be obtained in polynomial time.*

4 Results

4.1 Wikipedia

We downloaded February 2010 XML dumps of all available editions of Wikipedia, in total 272 editions that amount to 86.5 GB uncompressed. From these dumps we produced two datasets. Dataset A captures cross-lingual interwiki links between pages, in total 77.07 million undirected edges (146.76 million original links). Dataset B additionally includes 2.2 million redirect-based edges. Wikipedia deals with interwiki links to redirects transparently, however there are many redirects with titles that do not co-refer, e.g. redirects from members of a band to the band, or from aspects of a topic to the topic in general. We only included redirects in the following cases:

- the titles of redirect and redirect target match after Unicode NFKD normalization, diacritics removal, case conversion, and removal of punctuation characters
- the redirect uses certain templates or categories that indicate co-reference with the target (alternative names, abbreviations, etc.)

We treated them like reciprocal interwiki links by assigning them a weight of 2.

4.2 Application of Algorithm

The choice of distinctness assertion weights depends on how lenient we wish to be towards conceptual drift, allowing us to opt for more fine- or more coarse-grained distinctions. In our experiments, we decided to prefer fine-grained conceptual distinctions, and settled on a weight of 100.

We analysed over 20 million connected components in each dataset, checking for distinctness assertions. For the roughly 110,000 connected components with relevant distinctness assertions,

Algorithm 3.1 WDGS Approximation Algorithm

procedure SELECT($V, E, V', E', w, D_1, \ldots, D_n, l_1, \ldots, l_n$)
  solve linear program given by Definition 5 ▷ determine optimal fractional solution
  construct $G' = (V', E')$ ▷ extended graph (Definition 6)
  $C \leftarrow \{e \in E \mid w(e) = 0\}$ ▷ cut zero-weighted edges
  $U_i \leftarrow \bigcup_{j=1}^{l_i - 1} D_{i,j} \quad \forall i : w(D_i) = 0$ ▷ remove zero-weighted D_i
  while $\exists i, j, k > j, u \in D_{i,j}, v \in D_{i,k} : \mathrm{P}(\mathrm{v}_{i,u}, \mathrm{v}_{i,v}, E') \neq \emptyset$ **do** ▷ find unsatisfied assertion
    $S \leftarrow \emptyset$ ▷ set of nodes around which regions will be grown
    for all j in $1 \ldots l_i - 1$ **do** ▷ arbitrarily choose node from each $D_{i,j}$
      if $\exists v \in D_{i,j} : \mathrm{v}_{i,v} \in V'$ **then** $S \leftarrow S \cup \mathrm{v}_{i,v}$
    $D \leftarrow \{d(u,v) \leq \frac{1}{2} \mid u \in S, v \in V'\} \cup \{\frac{1}{2}\}$ ▷ set of distances
    choose ϵ such that $\forall d, d' \in D : 0 < \epsilon \ll |d - d'|$ ▷ infinitesimally small
    $r \leftarrow \operatorname{argmin}_{r = d - \epsilon:\ d \in D \setminus \{0\}} \frac{w(C(S,r))}{\hat{c}(S,r)}$ ▷ choose optimal radius (ties broken arbitrarily)
    $V' \leftarrow V' \setminus R(S,r)$ ▷ remove regions from G'
    $E' \leftarrow \{e \in E' \mid e \subseteq V'\}$
    $C \leftarrow C \cup (C(S,r) \cap E)$ ▷ update global solution
    for all i' in $1 \ldots n$ **do**
      $U_{i'} \leftarrow U_{i'} \cup \{v \mid (\mathrm{v}_{i',v}, v) \in C(S,r)\}$
      for all j in $1 \ldots l_{i'}$ **do** $D_{i',j} \leftarrow D_{i',j} \cap V'$ ▷ prune distinctness assertions
  return $(C, U_1, \ldots, U_n)$

we applied our algorithm, relying on the commercial CPLEX tool to solve the linear programs. In most cases, the LP solving took less than a second, however the LP sizes grow exponentially with the number of nodes and hence the time complexity increases similarly. In about 300 cases per dataset, CPLEX took too long and was automatically killed or the linear program was a priori deemed too large to complete in a short amount of time. For these cases, we adopted an alternative strategy described later on.

Table 1 provides the experimental results for the two datasets. Dataset B is more connected and thus has fewer connected components with more pairs of nodes asserted to be distinct by distinctness assertions. The LP given by Definition 5 provides fractional solutions that constitute lower bounds on the optimal solution (cf. also Lemma 5 in Appendix A), so the optimal solution cannot have a cost lower than the fractional LP solution. Table 1 shows that in practice, our algorithm achieves near-optimal results.

Table 1: Algorithm Results

	Dataset A	Dataset B
Connected components	23,356,027	21,161,631
– with distinctness assertions	112,857	113,714
– algorithm applied successfully	112,580	113,387
Distinctness assertions	380,694	379,724
Node pairs considered distinct	916,554	1,047,299
Lower bound on optimal cost	1,255,111	1,245,004
Cost of our solution	1,306,747	1,294,196
Factor	1.04	1.04
Edges to be deleted (undirected)	1,209,798	1,199,181
Nodes to be merged	603	573

4.3 Linguistic Adequacy

The near-optimal results of our algorithm apply with respect to our problem formalization, which aims at repairing the graph in a minimally invasive way. It may happen, however, that the graph's topology is misleading, and that in a specific case deleting many cross-lingual links to separate two entities is more appropriate than looking for a conservative way to separate them. This led us

to study the linguistic adequacy. Two annotators evaluated 200 randomly selected separated pairs from Dataset A consisting of an English and a German article, with an inter-annotator agreement (Cohen κ) of 0.656. Examples are given in Table 2. We obtained a precision of $87.97\% \pm 0.04\%$ (Wilson score interval) against the consensus annotation. Many of the errors are the result of articles having many inaccurate outgoing links, in which case they may be assigned to the wrong component. In other cases, we noted duplicate articles in Wikipedia.

Occasionally, we also observed differences in scope, where one article would actually describe two related concepts in a single page. Our algorithm will then either make a somewhat arbitrary assignment to the component of either the first or second concept, or the broader generalization of the two concepts becomes a separate, more general connected component.

4.4 Large Problem Instances

When problem instances become too large, the linear programs can become too unwieldy for linear optimization software to cope with on current hardware. In such cases, the graphs tend to be very sparsely connected, consisting of many smaller, more densely connected subgraphs. We thus investigated graph partitioning heuristics to decompose larger graphs into smaller parts that can more easily be handled with our algorithm. The METIS algorithms (Karypis and Kumar, 1998) can decompose graphs with hundreds of thousands of nodes almost instantly, but favour equally sized clusters over lower cut costs. We obtained partitionings with costs orders of magnitude lower using the heuristic by Dhillon et al. (2007).

4.5 Database of Named Entities

The partitioning heuristics allowed us to process all entries in the complete set of Wikipedia dumps and produce a clean output set of connected components where each Wikipedia article or category belongs to a connected component consisting of pages about the same entity or concept. We can regard these connected components as equivalence classes. This means that we obtain a large-scale multilingual database of named entities and their translations. We are also able to more safely transfer information cross-lingually between editions. For example, when an article a has a category c in the French Wikipedia, we can suggest the corresponding Indonesian category for the corresponding Indonesian article.

Moreover, we believe that this database will help extend resources like DBPedia and YAGO that to date have exclusively used the English Wikipedia as their repository of entities and classes. With YAGO's category heuristics, even entirely non-English connected components can be assigned a class in WordNet as long as at least one of the relevant categories has an English page. So, the French Wikipedia article on the Dutch schooner '*JR Tolkien*', despite the lack of a corresponding English article, can be assigned to the WordNet synset for '*ship*'. Using YAGO's plural heuristic to distinguish classes (Einstein *is a* physicist) from topic descriptors (Einstein *belongs to the topic* physics), we determined that over 4.8 million connected components can be linked to WordNet, greatly surpassing the 3.2 million articles covered by the English Wikipedia alone.

5 Related Work

A number of projects have used Wikipedia as a database of named entities (Ponzetto and Strube, 2007; Silberer et al., 2008). The most well-known are probably DBpedia (Auer et al., 2007), which serves as a hub in the Linked Data Web, Freebase[1], which combines human input and automatic extractors, and YAGO (Suchanek et al., 2007), which adds an ontological structure on top of Wikipedia's entities. Wikipedia has been used cross-lingually for cross-lingual IR (Nguyen et al., 2009), question answering (Ferrández et al., 2007) as well as for learning transliterations (Pasternack and Roth, 2009), among other things.

Mihalcea and Csomai (2007) have studied predicting new links within a single edition of Wikipedia. Sorg and Cimiano (2008) considered the problem of suggesting new cross-lingual links, which could be used as additional inputs in our problem. Adar et al. (2009) and Bouma et al. (2009) show how cross-lingual links can be used to propagate information from one Wikipedia's infoboxes to another edition.

Our aggregation consistency algorithm uses theoretical ideas put forward by researchers studying graph cuts (Leighton and Rao, 1999; Garg et al., 1996; Avidor and Langberg, 2007). Our problem setting is related to that of correlation clustering (Bansal et al., 2004), where a graph consist-

[1] `http://www.freebase.com/`

Table 2: Examples of separated concepts

English concept	German concept (translated)	Explanation
Coffee percolator	French Press	different types of brewing devices
Baqa-Jatt	Baqa al-Gharbiyye	Baqa-Jatt is a city resulting from a merger of Baqa al-Gharbiyye and Jatt
Leucothoe (plant)	Leucothea (Orchamos)	the second refers to a figure of Greek mythology
Old Belarusian language	Ruthenian language	the second is often considered slightly broader

ing of positively and negatively labelled similarity edges is clustered such that similar items are grouped together, however our approach is much more generic than conventional correlation clustering. Charikar et al. (2005) studied a variation of correlation clustering that is similar to WDGS, but since a negative edge would have to be added between each relevant pair of entities in a distinctness assertion, the approximation guarantee would only be $O(\log(n\ |V|^2))$. Minimally invasive repair operations on graphs have also been studied for graph similarity computation (Zeng et al., 2009), where two graphs are provided as input.

6 Conclusions and Future Work

We have presented an algorithmic framework for the problem of co-reference that produces consistent partitions by intelligently removing edges or allowing nodes to remain connected. This algorithm has successfully been applied to Wikipedia's cross-lingual graph, where we identified and eliminated surprisingly large numbers of inaccurate connections, leading to a large-scale multilingual register of names.

In future work, we would like to investigate how our algorithm behaves in extended settings, e.g. we can use heuristics to connect isolated, unconnected articles to likely candidates in other Wikipedias using weighted edges. This can be extended to include mappings from multiple languages to WordNet synsets, with the hope that the weights and link structure will then allow the algorithm to make the final disambiguation decision. Additional scenarios include dealing with co-reference on the Linked Data Web or mappings between thesauri. As such resources are increasingly being linked to Wikipedia and DBpedia, we believe that our techniques will prove useful in making mappings more consistent.

A Proofs

Proof (Theorem 1). We shall reduce the minimum multicut problem to WDGS. The hardness claims then follow from Chawla et al. (2005). Given a graph $G = (V, E)$ with a positive cost $c(e)$ for each $e \in E$, and a set $D = \{(s_i, t_i) \mid i = 1 \dots k\}$ of k demand pairs, our goal is to find a multicut M with respect to D with minimum total cost $\sum_{e \in M} c(e)$. We convert each demand pair (s_i, t_i) into a distinctness assertion $D_i = (\{s_i\}, \{t_i\})$ with weight $w(D_i) = 1 + \sum_{e \in E} c(e)$. An optimal WDGS solution $(C, U_1, \dots, U_k)$ with cost c then implies a multicut C with the same weight, because each $w(D_i) > \sum_{e \in E} c(e)$, so all demand pairs will be satisfied. C is a minimal multicut because any multicut C' with lower cost would imply a valid WDGS solution $(C', \emptyset, \dots, \emptyset)$ with a cost lower than the optimal one, which is a contradiction. □

Lemma 4. *The linear program given by Definition 5 enforces that for any i,j,k ≠ j,u ∈ $D_{i,j}$, $v \in D_{i,k}$, and any path $v_0, \dots, v_t$ with $v_0 = u$, $v_t = v$ we obtain $u_{i,u} + \sum_{l=0}^{t-1} d_{(v_l, v_{l+1})} + u_{i,v} \geq 1$. The integer linear program obtained by augmenting Definition 5 with integer constraints $d_e, u_{i,v}, p_{i,j,v} \in \{0, 1\}$ (for all applicable e, i, j, v) produces optimal solutions $(C, U_1, \dots, U_k)$ for* WDGS *problems, obtained as $C = (\{e \in E \mid d_e = 1\}$, $U_i = \{v \mid u_{i,v} = 1\}$.*

Proof. Without loss of generality, let us assume that $j < k$. The LP constraints give us $p_{i,j,v_t} \leq p_{i,j,v_{t-1}} + d_{(v_{t-1}, v_t)}, \dots, p_{i,j,v_1} \leq p_{i,j,v_0} + d_{(v_0, v_1)}$, as well as $p_{i,j,v_0} = u_{i,u}$ and $p_{i,j,v_t} + u_{i,v} \geq 1$. Hence $1 \leq p_{i,j,v_t} + u_{i,v} \leq u_{i,u} + \sum_{l=0}^{t-1} d_{(v_l, v_{l+1})} + u_{i,v}$.

With added integrality constraints, we obtain either $u \in U_i$, $v \in U_i$, or at least one edge along any path from u to v is cut, i.e. $\mathrm{P}(u, v, E \setminus C) = \emptyset$.

This proves that any ILP solution enduces a valid WDGS solution (Definition 2).

Clearly, the integer program's objective function minimizes $c(C, U_1, \ldots, U_n)$ (Definition 3) if $C = (\{e \in E \mid d_e = 1\}$, $U_i = \{v \mid u_{i,v} = 1\}$. To see that the solutions are optimal, it thus suffices to observe that any optimal WDGS solution $(C^*, U_1^*, \ldots, U_n^*)$ yields a feasible ILP solution $d_e = I_{C^*}(e)$, $u_{i,v} = I_{U_i^*}(v)$. □

Proof (Theorem 2). $r_i < \frac{1}{2}$ holds for any radius r_i chosen by the algorithm, so for any region $R(v_0, r)$ grown around a node v_0, and any two nodes u, v within that region, the triangle inequality gives us $d(u, v) \leq d(u, v_0) + d(v_0, v) < \frac{1}{2} + \frac{1}{2} = 1$ (maximal distance condition). At the same time, by Lemma 4 and Definition 6 for any $u \in D_{i,j}$, $v \in D_{i,k}$ $(j \neq k)$, we obtain $d(\mathrm{v}_{i,u}, \mathrm{v}_{i,v}) = d(\mathrm{v}_{i,u}, u) + d(u, v) + d(v, \mathrm{v}_{i,v}) \geq 1$. With the maximal distance condition above, this means that $\mathrm{v}_{i,u}$ and $\mathrm{v}_{i,v}$ cannot be in the same region. Hence u, v cannot be in the same region, unless the edge from $\mathrm{v}_{i,u}$ to u is cut (in which case u will be placed in U_i) or the edge from v to $\mathrm{v}_{i,v}$ is cut (in which case v will be placed in U_i). Since each region is separated from other regions via C, we obtain that $\forall i, j, k \neq j, u, v$: $u \in D_{i,j} \setminus U_i$, $v \in D_{i,k} \setminus U_i$ implies $\mathrm{P}(u, v, E \setminus C) = \emptyset$, so a valid solution is obtained. □

Lemma 5 (essentially due to Garg et al. (1996))**.** *For any i where $\exists j, k > j, u \in D_{i,j}, v \in D_{i,k}$: $\mathrm{P}(\mathrm{v}_{i,u}, \mathrm{v}_{i,v}, E') \neq \emptyset$ and $w(D_i) > 0$, there exists an r such that $w(C(S, r)) \leq 2\ln(nq+1)\,\hat{c}(S, r)$, $0 \leq r < \frac{1}{2}$ for any set S consisting of $\mathrm{v}_{i,v}$ nodes.*

Proof. Define $w(S, r) = \sum_{v \in S} w(C(v, r))$. We will prove that there exists an appropriate r with $w(C(S, r)) \leq w(S, r) \leq 2\ln(nq+1)\,\hat{c}(S, r)$. Assume, for reductio ad absurdum, that $\forall r \in [0, \frac{1}{2})$: $w(S, r) > 2\ln(nq+1)\hat{c}(S, r)$. As we expand the radius r, we note that $\hat{c}(S, r)\frac{d}{dr} = w(S, r)$ whereever $\hat{c}$ is differentiable with respect to r. There are only a finite number of points $r_1, \ldots, r_{l-1}$ in $(0, \frac{1}{2})$ where this is not the case (namely, when $\exists u \in S,\ v \in V' : d(u, v) = r_i$). Also note that $\hat{c}$ increases monotonically for increasing values of r, and that it is universally greater than zero (since there is a path between $\mathrm{v}_{i,u}$, $\mathrm{v}_{i,v}$). Set $r_0 = 0$, $r_l = \frac{1}{2}$ and choose ϵ such that $0 < \epsilon \ll \min\{r_{j+1} - r_j \mid j < l\}$. Our assumption then implies:

$$\sum_{j=1}^{l} \int_{r_{j-1}+\epsilon}^{r_j-\epsilon} \frac{w(S,r)}{\hat{c}(S,r)}\, dr > \left[\sum_{j=1}^{l} r_j - r_{j-1} - 2\epsilon\right] 2\ln(nq+1)$$

$$\sum_{j=1}^{l} \ln \hat{c}(S, r_j - \epsilon) - \ln \hat{c}(S, r_{j-1} + \epsilon) > \left(\tfrac{1}{2} - 2l\epsilon\right) 2\ln(nq+1)$$

$$\ln \hat{c}(S, \tfrac{1}{2} - \epsilon) - \ln \hat{c}(S, 0) > (1 - 4l\epsilon)\ln(nq+1)$$

$$\frac{\hat{c}(S, \frac{1}{2}-\epsilon)}{\hat{c}(S,0)} > (nq+1)^{1-4l\epsilon}$$

$$\hat{c}(S, \tfrac{1}{2} - \epsilon) > (nq+1)^{1-4l\epsilon}\hat{c}(S, 0)$$

For small ϵ, the right term can get arbitrarily close to $(nq+1)\hat{c}(S, 0) \geq \hat{c}_{\mathrm{f}} + \hat{c}(S, 0)$, which is strictly larger than $\hat{c}(S, \frac{1}{2} - \epsilon)$ no matter how small ϵ becomes, so the initial assumption is false. □

Proof (Theorem 3). Let S_i, r_i denote the set S and radius r chosen in particular iterations, and c_i the corresponding costs incurred: $c_i = w(C(S_i, r) \cap E) + |U_i| w(D_i) = w(C(D_i, r))$. Note that any r_i chosen by the algorithm will in fact fulfil the criterion described by Lemma 5, because r_i is chosen to minimize the ratio between the two terms, and the minimizing $r \in [0, \frac{1}{2})$ must be among the r considered by the algorithm ($w(C(D_i, r))$ only changes at one of those points, so the minimum is reached by approaching the points from the left). Hence, we obtain $c_i \leq 2\ln(n+1)\hat{c}(S_i, r_i)$. For our global solution, note that there is no overlap between the regions chosen within an iteration, since regions have a radius strictly smaller than $\frac{1}{2}$, while $\mathrm{v}_{i,u}$, $\mathrm{v}_{i,v}$ for $u \in D_{i,j}$, $v \in D_{i,k}$, $j \neq k$ have a distance of at least 1. Nor is there any overlap between regions from different iterations, because in each iteration the selected regions are removed from G'. Globally, we therefore obtain $c(C, U_1, \ldots, U_n) = \sum_i c_i < 2\ln(nq+1) \sum_i \hat{c}(S_i, r_i) \leq 2\ln(nq+1)2\hat{c}_{\mathrm{f}}$ (observe that $i \leq nq$). Since $\hat{c}_{\mathrm{f}}$ is the objective score for the fractional LP relaxation solution of the WDGS ILP (Lemma 4), we obtain $\hat{c}_{\mathrm{f}} \leq c(C^*, U_1^*, \ldots, U_n^*)$, and thus $c(C, U_1, \ldots, U_n) < 4\ln(n+1)c(C^*, U_1^*, \ldots, U_n^*)$.

To obtain a solution in polynomial time, note that the LP size is polynomial with respect to nq and may be solved using a polynomial algorithm (Karmarkar, 1984). The subsequent steps run in $O(nq)$ iterations, each growing up to $|V|$ regions using $O(|V|^2)$ uniform cost searches. □

References

Eytan Adar, Michael Skinner, and Daniel S. Weld. 2009. Information arbitrage across multi-lingual Wikipedia. In Ricardo A. Baeza-Yates, Paolo Boldi, Berthier A. Ribeiro-Neto, and Berkant Barla Cambazoglu, editors, *Proceedings of the 2nd International Conference on Web Search and Web Data Mining, WSDM 2009*, pages 94–103. ACM.

Sören Auer, Chris Bizer, Jens Lehmann, Georgi Kobilarov, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In Aberer et al., editor, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11–15, 2007*, Lecture Notes in Computer Science 4825. Springer.

Adi Avidor and Michael Langberg. 2007. The multi-multiway cut problem. *Theoretical Computer Science*, 377(1-3):35–42.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine Learning*, 56(1-3):89–113.

Gosse Bouma, Sergio Duarte, and Zahurul Islam. 2009. Cross-lingual alignment and completion of Wikipedia templates. In *CLIAWS3 '09: Proceedings of the Third International Workshop on Cross Lingual Information Access*, pages 21–29, Morristown, NJ, USA. Association for Computational Linguistics.

Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383.

Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. 2005. On the hardness of approximating multicut and sparsest-cut. In *In Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 144–153.

Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors. a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957.

Sergio Ferrández, Antonio Toral, Óscar Ferrández, Antonio Ferrández, and Rafael Muñoz. 2007. Applying Wikipedia's multilingual knowledge to cross-lingual question answering. In *NLDB*, pages 352–363.

Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. 1996. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing (SICOMP)*, 25:698–707.

Narendra Karmarkar. 1984. A new polynomial-time algorithm for linear programming. In *STOC '84: Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, New York, NY, USA. ACM.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392.

Subhash Khot. 2002. On the power of unique 2-prover 1-round games. In *STOC '02: Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 767–775, New York, NY, USA. ACM.

Tom Leighton and Satish Rao. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pages 233–242, New York, NY, USA. ACM.

D. Nguyen, A. Overwijk, C. Hauff, R.B. Trieschnigg, D. Hiemstra, and F.M.G. Jong de. 2009. WikiTranslate: query translation for cross-lingual information retrieval using only Wikipedia. In Carol Peters, Thomas Deselaers, Nicola Ferro, and Julio Gonzalo, editors, *Evaluating Systems for Multilingual and Multimodal Information Access*, Lecture Notes in Computer Science 5706, pages 58–65.

Jeff Pasternack and Dan Roth. 2009. Learning better transliterations. In *CIKM '09: Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 177–186, New York, NY, USA. ACM.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *AAAI 2007: Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1440–1445. AAAI Press.

Carina Silberer, Wolodja Wentland, Johannes Knopp, and Matthias Hartung. 2008. Building a multilingual lexical resource for named entity disambiguation, translation and transliteration. In European, editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Philipp Sorg and Philipp Cimiano. 2008. Enriching the crosslingual link structure of Wikipedia - a classification-based approach. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artifical Intelligence*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International World Wide Web conference, WWW*, New York, NY, USA. ACM Press.

Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. 2009. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36.

Bucking the Trend: Large-Scale Cost-Focused Active Learning for Statistical Machine Translation

Michael Bloodgood
Human Language Technology
Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
bloodgood@jhu.edu

Chris Callison-Burch
Center for Language and
Speech Processing
Johns Hopkins University
Baltimore, MD 21211
ccb@cs.jhu.edu

Abstract

We explore how to improve machine translation systems by adding more translation data in situations where we already have substantial resources. The main challenge is how to buck the trend of diminishing returns that is commonly encountered. We present an active learning-style data solicitation algorithm to meet this challenge. We test it, gathering annotations via Amazon Mechanical Turk, and find that we get an order of magnitude increase in performance rates of improvement.

1 Introduction

Figure 1 shows the learning curves for two state of the art statistical machine translation (SMT) systems for Urdu-English translation. Observe how the learning curves rise rapidly at first but then a trend of diminishing returns occurs: put simply, the curves flatten.

This paper investigates whether we can buck the trend of diminishing returns, and if so, how we can do it effectively. Active learning (AL) has been applied to SMT recently (Haffari et al., 2009; Haffari and Sarkar, 2009) but they were interested in starting with a tiny seed set of data, and they stopped their investigations after only adding a relatively tiny amount of data as depicted in Figure 1.

In contrast, we are interested in applying AL when a large amount of data already exists as is the case for many important lanuage pairs. We develop an AL algorithm that focuses on keeping annotation costs (measured by time in seconds) low. It succeeds in doing this by only soliciting translations for parts of sentences. We show that this gets a savings in human annotation time above and beyond what the reduction in # words annotated would have indicated by a factor of about three and speculate as to why.

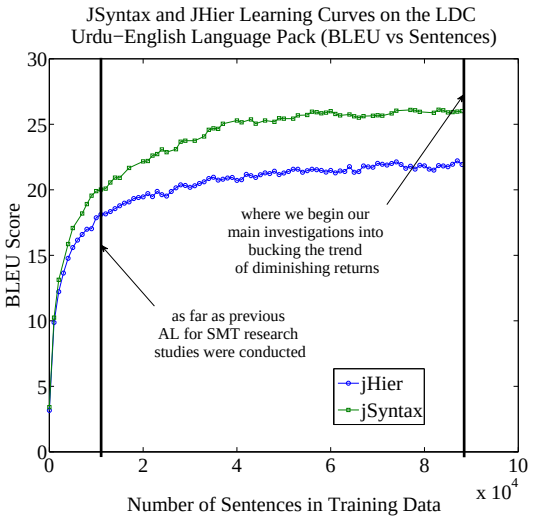


Figure 1: Syntax-based and Hierarchical Phrase-Based MT systems' learning curves on the LDC Urdu-English language pack. The x-axis measures the number of sentence pairs in the training data. The y-axis measures BLEU score. Note the diminishing returns as more data is added. Also note how relatively early on in the process previous studies were terminated. In contrast, the focus of our main experiments doesn't even begin until much higher performance has already been achieved with a period of diminishing returns firmly established.

We conduct experiments for Urdu-English translation, gathering annotations via Amazon Mechanical Turk (MTurk) and show that we can indeed buck the trend of diminishing returns, achieving an order of magnitude increase in the rate of improvement in performance.

Section 2 discusses related work; Section 3 discusses preliminary experiments that show the guiding principles behind the algorithm we use; Section 4 explains our method for soliciting new translation data; Section 5 presents our main results; and Section 6 concludes.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 854–864,
Uppsala, Sweden, 11-16 July 2010.

2 Related Work

Active learning has been shown to be effective for improving NLP systems and reducing annotation burdens for a number of NLP tasks (see, e.g., (Hwa, 2000; Sassano, 2002; Bloodgood and Vijay-Shanker, 2008; Bloodgood and Vijay-Shanker, 2009b; Mairesse et al., 2010; Vickrey et al., 2010)). The current paper is most highly related to previous work falling into three main areas: use of AL when large corpora already exist; cost-focused AL; and AL for SMT.

In a sense, the work of Banko and Brill (2001) is closely related to ours. Though their focus is mainly on investigating the performance of learning methods on giant corpora many orders of magnitude larger than previously used, they do lay out how AL might be useful to apply to acquire data to augment a large set cheaply because they recognize the problem of diminishing returns that we discussed in Section 1.

The second area of work that is related to ours is previous work on AL that is cost-conscious. The vast majority of AL research has not focused on accurate cost accounting and a typical assumption is that each annotatable has equal annotation cost. An early exception in the AL for NLP field was the work of Hwa (2000), which makes a point of using # of brackets to measure cost for a syntactic analysis task instead of using # of sentences. Another relatively early work in our field along these lines was the work of Ngai and Yarowsky (2000), which measured actual times of annotation to compare the efficacy of rule writing versus annotation with AL for the task of BaseNP chunking. Osborne and Baldridge (2004) argued for the use of discriminant cost over unit cost for the task of Head Phrase Structure Grammar parse selection. King et al. (2004) design a robot that tests gene functions. The robot chooses which experiments to conduct by using AL and takes monetary costs (in pounds sterling) into account during AL selection and evaluation. Unlike our situation for SMT, their costs are all known beforehand because they are simply the cost of materials to conduct the experiments, which are already known to the robot. Hachey et al. (2005) showed that selectively sampled examples for an NER task took longer to annotate and had lower inter-annotator agreement. This work is related to ours because it shows that how examples are selected can impact the cost of annotation, an idea we turn around to use for our advantage when developing our data selection algorithm. Haertel et al. (2008) emphasize measuring costs carefully for AL for POS tagging. They develop a model based on a user study that can estimate the time required for POS annotating. Kapoor et al. (2007) assign costs for AL based on message length for a voicemail classification task. In contrast, we show for SMT that annotation times do not scale according to length in words and we show our method can achieve a speedup in annotation time above and beyond what the reduction in words would indicate. Tomanek and Hahn (2009) measure cost by # of tokens for an NER task. Their AL method only solicits labels for parts of sentences in the interest of reducing annotation effort. Along these lines, our method is similar in the respect that we also will only solicit annotation for parts of sentences, though we prefer to measure cost with time and we show that time doesn't track with token length for SMT.

Haffari et al. (2009), Haffari and Sarkar (2009), and Ambati et al. (2010) investigate AL for SMT. There are two major differences between our work and this previous work. One is that our intended use cases are very different. They deal with the more traditional AL setting of starting from an extremely small set of seed data. Also, by SMT standards, they only add a very tiny amount of data during AL. All their simulations top out at 10,000 sentences of labeled data and the models learned have relatively low translation quality compared to the state of the art.

On the other hand, in the current paper, we demonstrate how to apply AL in situations where we already have large corpora. Our goal is to buck the trend of diminishing returns and use AL to add data to build some of the highest-performing MT systems in the world while keeping annotation costs low. See Figure 1 from Section 1, which contrasts where (Haffari et al., 2009; Haffari and Sarkar, 2009) *stop* their investigations with where we *begin* our studies.

The other major difference is that (Haffari et al., 2009; Haffari and Sarkar, 2009) measure annotation cost by # of sentences. In contrast, we bring to light some potential drawbacks of this practice, showing it can lead to different conclusions than if other annotation cost metrics are used, such as time and money, which are the metrics that we use.

3 Simulation Experiments

Here we report on results of simulation experiments that help to illustrate and motivate the design decisions of the algorithm we present in Section 4. We use the Urdu-English language pack[1] from the Linguistic Data Consortium (LDC), which contains ≈ 88000 Urdu-English sentence translation pairs, amounting to ≈ 1.7 million Urdu words translated into English. All experiments in this paper evaluate on a genre-balanced split of the NIST2008 Urdu-English test set. In addition, the language pack contains an Urdu-English dictionary consisting of ≈ 114000 entries. In all the experiments, we use the dictionary at every iteration of training. This will make it harder for us to show our methods providing substantial gains since the dictionary will provide a higher base performance to begin with. However, it would be artificial to ignore dictionary resources when they exist.

We experiment with two translation models: hierarchical phrase-based translation (Chiang, 2007) and syntax augmented translation (Zollmann and Venugopal, 2006), both of which are implemented in the Joshua decoder (Li et al., 2009). We hereafter refer to these systems as jHier and jSyntax, respectively.

We will now present results of experiments with different methods for growing MT training data. The results are organized into three areas of investigations:

1. annotation costs;
2. managing uncertainty; and
3. how to automatically detect when to stop soliciting annotations from a pool of data.

3.1 Annotation Costs

We begin our cost investigations with four simple methods for growing MT training data: random, shortest, longest, and *VocabGrowth* sentence selection. The first three methods are self-explanatory. *VocabGrowth* (hereafter *VG*) selection is modeled after the best methods from previous work (Haffari et al., 2009; Haffari and Sarkar, 2009), which are based on preferring sentences that contain phrases that occur frequently in unlabeled data and infrequently in the so-far labeled data. Our *VG* method selects sentences for translation that contain n-grams (for n in {1,2,3,4}) that do not occur at all in our so-far labeled data. We call an n-gram "covered" if it occurs at least once in our so-far labeled data. *VG* has a preference for covering frequent n-grams before covering infrequent n-grams. The *VG* method is depicted in Figure 2.

Init:
Go through all available training data (labeled and unlabeled) and obtain frequency counts for every n-gram (n in $\{1, 2, 3, 4\}$) that occurs.
$sortedNGrams \leftarrow$ Sort n-grams by frequency in descending order.

Loop
until stopping criterion (see Section 3.3) is met
1. $trigger \leftarrow$ Go down $sortedNGrams$ list and find the first n-gram that isn't covered in the so far labeled training data.
2. $selectedSentence \leftarrow$ Find a sentence that contains $trigger$.
3. Remove $selectedSentence$ from unlabeled data and add it to labeled training data.

End Loop

Figure 2: The *VG* sentence selection algorithm

Figure 3 shows the learning curves for both jHier and jSyntax for *VG* selection and random selection. The y-axis measures BLEU score (Papineni et al., 2002),which is a fast automatic way of measuring translation quality that has been shown to correlate with human judgments and is perhaps the most widely used metric in the MT community. The x-axis measures the number of sentence translation pairs in the training data. The *VG* curves are cut off at the point at which the stopping criterion in Section 3.3 is met. From Figure 3 it might appear that *VG* selection is better than random selection, achieving higher-performing systems with fewer translations in the labeled data.

However, it is important to take care when measuring annotation costs (especially for relatively complicated tasks such as translation). Figure 4 shows the learning curves for the same systems and selection methods as in Figure 3 but now the x-axis measures the number of foreign words in the training data. The difference between *VG* and random selection now appears smaller.

For an extreme case, to illustrate the ramifica-

[1] LDC Catalog No.: LDC2006E110.

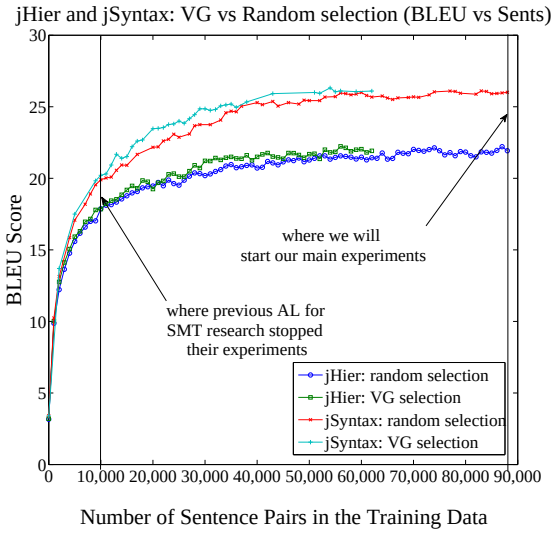


Figure 3: Random vs *VG* selection. The x-axis measures the number of sentence pairs in the training data. The y-axis measures BLEU score.

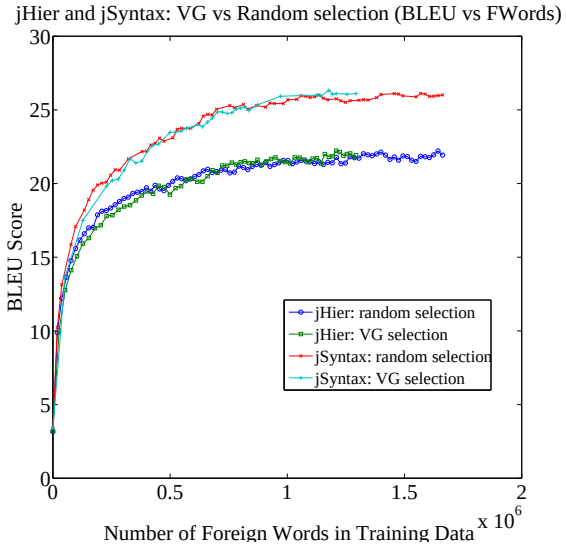


Figure 4: Random vs *VG* selection. The x-axis measures the number of foreign words in the training data. The y-axis measures BLEU score.

tions of measuring translation annotation cost by # of sentences versus # of words, consider Figures 5 and 6. They both show the same three selection methods but Figure 5 measures the x-axis by # of sentences and Figure 6 measures by # of words. In Figure 5, one would conclude that shortest is a far inferior selection method to longest but in Figure 6 one would conclude the opposite.

Measuring annotation time and cost in dollars are probably the most important measures of annotation cost. We can't measure these for the simulated experiments but we will use time (in seconds) and money (in US dollars) as cost measures in Section 5, which discusses our non-simulated AL experiments. If # sentences or # words track these other more relevant costs in predictable known relationships, then it would suffice to measure # sentences or # words instead. But it's clear that different sentences can have very different annotation time requirements according to how long and complicated they are so we will not use # sentences as an annotation cost any more. It is not as clear how # words tracks with annotation time. In Section 5 we will present evidence showing that time per word can vary considerably and also show a method for soliciting annotations that reduces time per word by nearly a factor of three.

As it is prudent to evaluate using accurate cost accounting, so it is also prudent to develop new AL algorithms that take costs carefully into account. Hence, reducing annotation time burdens instead of the # of sentences translated (which might be quite a different thing) will be a cornerstone of the algorithm we describe in Section 4.

3.2 Managing Uncertainty

One of the most successful of all AL methods developed to date is uncertainty sampling and it has been applied successfully many times (e.g.,(Lewis and Gale, 1994; Tong and Koller, 2002)). The intuition is clear: much can be learned (potentially) if there is great uncertainty. However, with MT being a relatively complicated task (compared with binary classification, for example), it might be the case that the uncertainty approach has to be re-considered. If words have never occurred in the training data, then uncertainty can be expected to be high. But we are concerned that if a sentence is translated for which (almost) no words have been seen in training yet, though uncertainty will be high (which is usually considered good for AL), the word alignments may be incorrect and then subsequent learning from that translation pair will be severely hampered.

We tested this hypothesis and Figure 7 shows empirical evidence that it is true. Along with *VG*, two other selection methods' learning curves are charted in Figure 7: mostNew, which prefers to select those sentences which have the largest # of unseen words in them; and moderateNew, which aims to prefer sentences that have a moderate # of unseen words, preferring sentences with $\approx$ ten

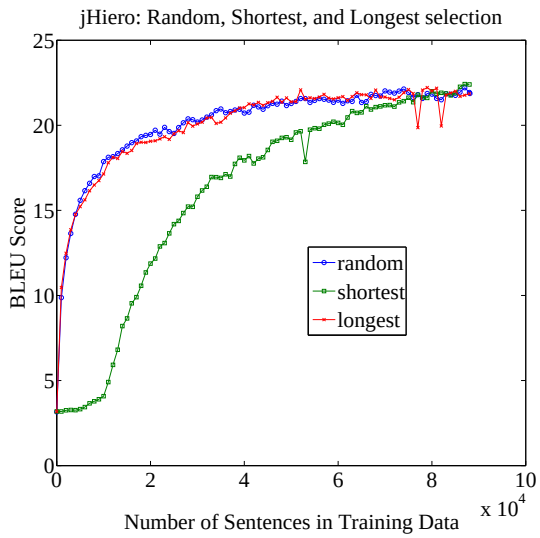


Figure 5: Random vs Shortest vs Longest selection. The x-axis measures the number of sentence pairs in the training data. The y-axis measures BLEU score.

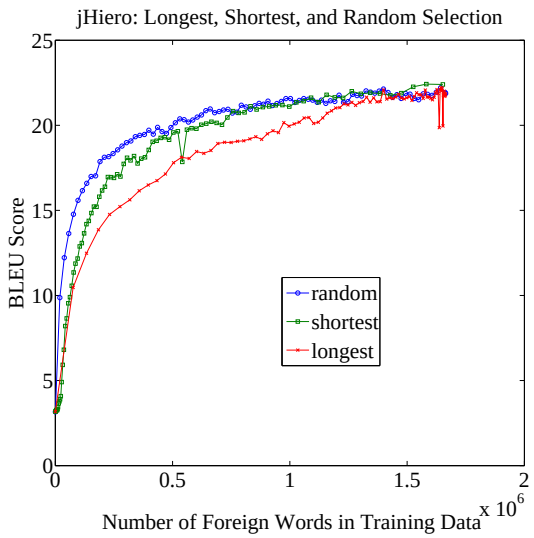


Figure 6: Random vs Shortest vs Longest selection. The x-axis measures the number of foreign words in the training data. The y-axis measures BLEU score.

unknown words in them. One can see that mostNew underperforms *VG*. This could have been due to *VG*'s frequency component, which mostNew doesn't have. But moderateNew also doesn't have a frequency preference so it is likely that mostNew winds up overwhelming the MT training system, word alignments are incorrect, and less is learned as a result. In light of this, the algorithm we develop in Section 4 will be designed to avoid this word alignment danger.

3.3 Automatic Stopping

The problem of automatically detecting when to stop AL is a substantial one, discussed at length in the literature (e.g., (Bloodgood and Vijay-Shanker, 2009a; Schohn and Cohn, 2000; Vlachos, 2008)). In our simulation, we stop *VG* once all n-grams (n in {1,2,3,4}) have been covered. Though simple, this stopping criterion seems to work well as can be seen by where the curve for *VG* is cut off in Figures 3 and 4. It stops after 1,293,093 words have been translated, with jHier's BLEU=21.92 and jSyntax's BLEU=26.10 at the stopping point. The ending BLEU scores (with the full corpus annotated) are 21.87 and 26.01 for jHier and jSyntax, respectively. So our stopping criterion saves 22.3% of the annotation (in terms of words) and actually achieves slightly higher BLEU scores than if all the data were used. Note: this "less is more" phenomenon has been commonly observed in AL settings (e.g., (Bloodgood and Vijay-Shanker, 2009a; Schohn and Cohn, 2000)).

4 Highlighted N-Gram Method

In this section we describe a method for soliciting human translations that we have applied successfully to improving translation quality in real (not simulated) conditions. We call the method the *Highlighted N-Gram* method, or *HNG*, for short. *HNG* solicits translations only for trigger n-grams and not for entire sentences. We provide sentential context, highlight the trigger n-gram that we want translated, and ask for a translation of just the highlighted trigger n-gram. *HNG* asks for translations for triggers in the same order that the triggers are encountered by the algorithm in Figure 2. A screenshot of our interface is depicted in Figure 8. The same stopping criterion is used as was used in the last section. When the stopping criterion becomes true, it is time to tap a new unlabeled pool of foreign text, if available.

Our motivations for soliciting translations for only parts of sentences are twofold, corresponding to two possible cases. Case one is that a translation model learned from the so-far labeled data will be able to translate most of the non-trigger words in the sentence correctly. Thus, by asking a human to translate only the trigger words, we avoid wasting human translation effort. (We will show in

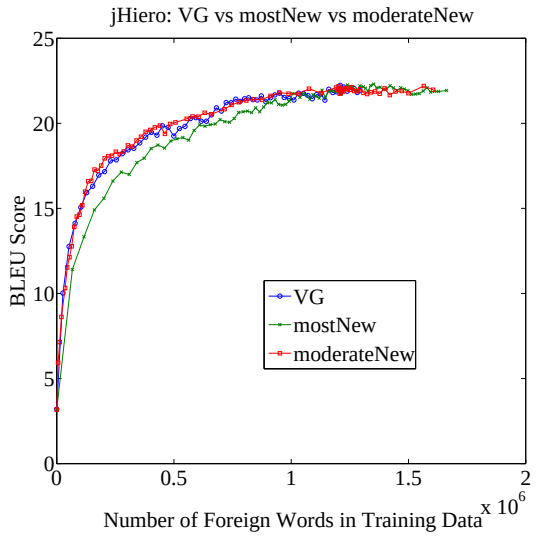


Figure 7: *VG* vs MostNew vs ModerateNew selection. The x-axis measures the number of sentence pairs in the training data. The y-axis measures BLEU score.

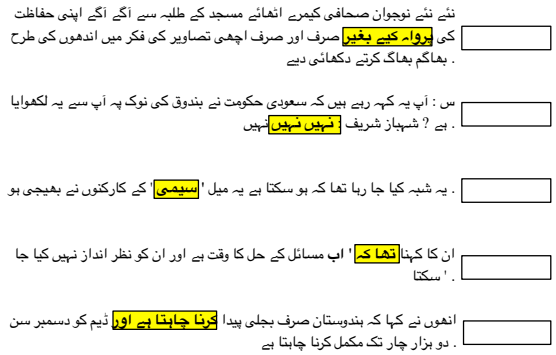


Figure 8: Screenshot of the interface we used for soliciting translations for triggers.

the next section that we even get a much larger speedup above and beyond what the reduction in number of translated words would give us.) Case two is that a translation model learned from the so-far labeled data will (in addition to not being able to translate the trigger words correctly) also not be able to translate most of the non-trigger words correctly. One might think then that this would be a great sentence to have translated because the machine can potentially learn a lot from the translation. Indeed, one of the overarching themes of AL research is to query examples where uncertainty is greatest. But, as we showed evidence for in the last section, for the case of SMT, too much uncertainty could in a sense overwhelm the machine and it might be better to provide new training data in a more gradual manner. A sentence with large #s of unseen words is likely to get word-aligned incorrectly and then learning from that translation could be hampered. By asking for a translation of only the trigger words, we expect to be able to circumvent this problem in large part.

The next section presents the results of experiments that show that the *HNG* algorithm is indeed practically effective. Also, the next section analyzes results regarding various aspects of *HNG*'s behavior in more depth.

5 Experiments and Discussion

5.1 General Setup

We set out to see whether we could use the *HNG* method to achieve translation quality improvements by gathering additional translations to add to the training data of the entire LDC language pack, including its dictionary. In particular, we wanted to see if we could achieve translation improvements on top of already state-of-the-art performing systems trained already on the *entire* LDC corpus. Note that at the outset this is an ambitious endeavor (recall the flattening of the curves in Figure 1 from Section 1).

Snow et al. (2008) explored the use of the Amazon Mechanical Turk (MTurk) web service for gathering annotations for a variety of natural language processing tasks and recently MTurk has been shown to be a quick, cost-effective way to gather Urdu-English translations (Bloodgood and Callison-Burch, 2010). We used the MTurk web service to gather our annotations. Specifically, we first crawled a large set of BBC articles on the internet in Urdu and used this as our unlabeled pool from which to gather annotations. We applied the *HNG* method from Section 4 to determine what to post on MTurk for workers to translate.[2] We gathered 20,580 n-gram translations for which we paid $0.01 USD per translation, giving us a total cost of $205.80 USD. We also gathered 1632 randomly chosen Urdu sentence translations as a control set, for which we paid $0.10 USD per sentence translation.[3]

[2]For practical reasons we restricted ourselves to not considering sentences that were longer than 60 Urdu words, however.

[3]The prices we paid were not market-driven. We just chose prices we thought were reasonable. In hindsight, given how much quicker the phrase translations are for people we could have had a greater disparity in price.

5.2 Accounting for Translation Time

MTurk returns with each assignment the "WorkTimeInSeconds." This is the amount of time between when a worker accepts an assignment and when the worker submits the completed assignment. We use this value to estimate annotation times.[4]

Figure 9 shows *HNG* collection versus random collection from MTurk. The x-axis measures the number of seconds of annotation time. Note that *HNG* is more effective. A result that may be particularly interesting is that *HNG* results in a time speedup by more than just the reduction in translated words would indicate. The average time to translate a word of Urdu with the sentence postings to MTurk was 32.92 seconds. The average time to translate a word with the *HNG* postings to MTurk was 11.98 seconds. This is nearly three times faster. Figure 10 shows the distribution of speeds (in seconds per word) for *HNG* postings versus complete sentence postings. Note that the *HNG* postings consistently result in faster translation speeds than the sentence postings[5].

We hypothesize that this speedup comes about because when translating a full sentence, there's the time required to examine each word and translate them in some sense (even if not one-to-one) and then there is an extra significant overhead time to put it all together and synthesize into a larger sentence translation. The factor of three speedup is evidence that this overhead is significant effort compared to just quickly translating short n-grams from a sentence. This speedup is an additional benefit of the *HNG* approach.

5.3 Bucking the Trend

We gathered translations for ≈ 54,500 Urdu words via the use of *HNG* on MTurk. This is a relatively small amount, ≈ 3% of the LDC corpus. Figure 11 shows the performance when we add this training data to the LDC corpus. The rectangle around the last 700,000 words of the LDC data is wide and short (it has a height of 0.9 BLEU points and a width of 700,000 words) but the rectangle around the newly added translations is narrow and tall (a height of 1 BLEU point and a width of 54,500 words). Visually, it appears we are succeeding in bucking the trend of diminishing returns. We further confirmed this by running a least-squares linear regression on the points of the last 700,000 words annotated in the LDC data and also for the points in the new data that we acquired via MTurk for $205.80 USD. We find that the slope fit to our new data is 6.6245E-06 BLEU points per Urdu word, or 6.6245 BLEU points for a million Urdu words. The slope fit to the LDC data is only 7.4957E-07 BLEU points per word, or only 0.74957 BLEU points for a million words. This is already an order of magnitude difference that would make the difference between it being worth adding more data and not being worth it; and this is leaving aside the added time speedup that our method enjoys.

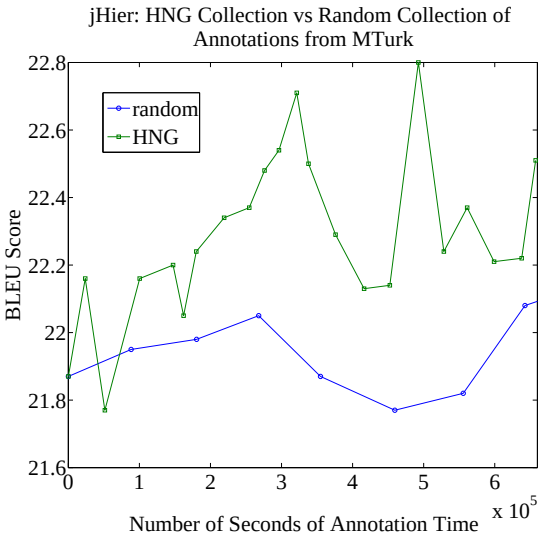


Figure 9: *HNG* vs Random collection of new data via MTurk. y-axis measures BLEU. x-axis measures annotation time in seconds.

Still, we wondered why we could not have raised BLEU scores even faster. The main hurdle seems to be one of coverage. Of the 20,580 n-grams we collected, only 571 (i.e., 2.77%) of them ever even occur in the test set.

5.4 Beyond BLEU Scores

BLEU is an imperfect metric (Callison-Burch et al., 2006). One reason is that it rates all ngram

[4]It's imperfect because of network delays and if a person is multitasking or pausing between their accept and submit times. Nonetheless, the times ought to be better estimates as they are taken over larger samples.

[5]The average speed for the *HNG* postings seems to be slower than the histogram indicates. This is because there were a few extremely slow outlier speeds for a handful of *HNG* postings. These are almost certainly not cases when the turker is working continuously on the task and so the average speed we computed for the *HNG* postings might be slower than the actual speed and hence the true speedup may even be faster than indicated by the difference between the average speeds we reported.

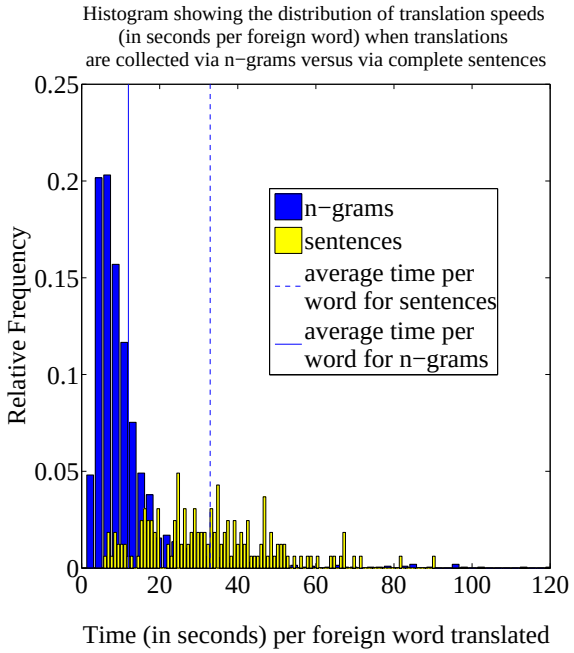


Figure 10: Distribution of translation speeds (in seconds per word) for *HNG* postings versus complete sentence postings. The y-axis measures relative frequency. The x-axis measures translation speed in seconds per word (so farther to the left is faster).

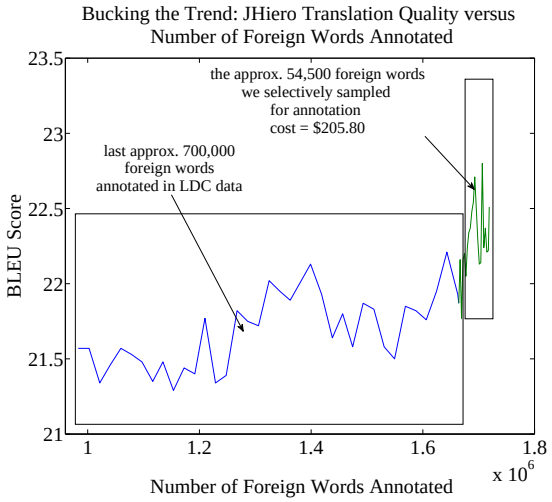


Figure 11: Bucking the trend: performance of *HNG*-selected additional data from BBC web crawl data annotated via Amazon Mechanical Turk. y-axis measures BLEU. x-axis measures number of words annotated.

- Learned phrase: "ناگالینڈ" means "nagaland"
- The "Before System" translation:
  - according to police the number of hundreds of ناگالینڈarmed tribesmen in the گلیکیof assam and fell سیبساset fire in three villages .
- The "After System" translation:
  - according to police the number of hundreds of nagaland armed tribesmen in the گلیکیof assam and fell سیبساset fire in three villages .
- Reference Translation:
  - according to the police , hundreds of armed nagaland tribesmen set on fire three villages in galleci and sebsagarh areas of assam .

Figure 12: Example of strategy working.

mismatches equally although some are much more important than others. Another reason is it's not intuitive what a gain of x BLEU points means in practice. Here we show some concrete example translations to show the types of improvements we're achieving and also some examples which suggest improvements we can make to our AL selection algorithm in the future. Figure 12 shows a prototypical example of our system working.

Figure 13 shows an example where the strategy is working partially but not as well as it might. The Urdu phrase was translated by turkers as "gowned veil". However, since the word aligner just aligns the word to "gowned", we only see "gowned" in our output. This prompts a number of discussion points. First, the 'after system' has better translations but they're not rewarded by BLEU scores because the references use the words 'burqah' or just 'veil' without 'gowned'. Second, we hypothesize that we may be able to see improvements by overriding the automatic alignment software whenever we obtain a many-to-one or one-to-many (in terms of words) translation for one of our trigger phrases. In such cases, we'd like to make sure that every word on the 'many' side is aligned to the single word on the 'one' side. For example, we would force both 'gowned' and 'veil' to be aligned to the single Urdu word instead of allowing the automatic aligner to only align 'gowned'.

Figure 14 shows an example where our "before" system already got the translation correct without the need for the additional phrase translation. This is because though the "before" system had never seen the Urdu expression for "12 May", it had seen the Urdu words for "12" and "May" in isolation and was able to successfully compose them. An area of future work is to use the "before" system to determine such cases automatically and avoid asking humans to provide translations in such cases.

- Learned phrase: "برقعے" means "gowned veil"
- The "Before System" translation:
  - ' in برقعےmaulana in برقعےgeneral is not islam in ! برقعے
- The "After System" translation:
  - ' is in gowned , maulana gowned , in general is not islam in gowned !
- Reference translations:
  - in burqah , there is the maulana . in burqah , there is the general . in burqah , there is no islam .
  - molana is under veil , and the general is under veil , but islam is not under veil .

Figure 13: Example showing where we can improve our selection strategy.

- Example: "بارہ مئی" means "12 may"
  - The "Before System" translation:
    - all party conference in speeches during the meeting two days of karachi on 12 may in connection with the incidents , was strongly criticized the mqm .
  - The "After System" translation:
    - during the meeting two days of all party conference held in karachi on 12 may in the speeches about the incidents were strongly criticized the mqm .

Figure 14: Example showing where we can improve our selection strategy.

6 Conclusions and Future Work

We succeeded in bucking the trend of diminishing returns and improving translation quality while keeping annotation costs low. In future work we would like to apply these ideas to domain adaptation (say, general-purpose MT system to work for scientific domain such as chemistry). Also, we would like to test with more languages, increase the amount of data we can gather, and investigate stopping criteria further. Also, we would like to investigate increasing the efficiency of the selection algorithm by addressing issues such as the one raised by the 12 May example presented earlier.

Acknowledgements

This work was supported by the Johns Hopkins University Human Language Technology Center of Excellence. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

References

Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July. Association for Computational Linguistics.

Michael Bloodgood and Chris Callison-Burch. 2010. Using mechanical turk to build machine translation evaluation sets. In *Proceedings of the Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, Los Angeles, California, June. Association for Computational Linguistics.

Michael Bloodgood and K Vijay-Shanker. 2008. An approach to reducing annotation costs for bionlp. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 104–105, Columbus, Ohio, June. Association for Computational Linguistics.

Michael Bloodgood and K Vijay-Shanker. 2009a. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 39–47, Boulder, Colorado, June. Association for Computational Linguistics.

Michael Bloodgood and K Vijay-Shanker. 2009b. Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 137–140, Boulder, Colorado, June. Association for Computational Linguistics.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, Trento, Italy.

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

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 144–151, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. 2008. Assessing the

costs of sampling methods in active learning for annotation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 65–68, Columbus, Ohio, June. Association for Computational Linguistics.

Gholamreza Haffari and Anoop Sarkar. 2009. Active learning for multilingual statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 181–189, Suntec, Singapore, August. Association for Computational Linguistics.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423, Boulder, Colorado, June. Association for Computational Linguistics.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In Hinrich Schütze and Keh-Yih Su, editors, *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 45–53. Association for Computational Linguistics, Somerset, New Jersey.

Ashish Kapoor, Eric Horvitz, and Sumit Basu. 2007. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 877–882.

Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. 2004. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 15 January.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March. Association for Computational Linguistics.

Francois Mairesse, Milica Gasic, Filip Jurcicek, Simon Keizer, Jorge Prombonas, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, July. Association for Computational Linguistics.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 89–96, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Manabu Sassano. 2002. An empirical study of active learning with support vector machines for japanese word segmentation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 505–512, Morristown, NJ, USA. Association for Computational Linguistics.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii, October. Association for Computational Linguistics.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1039–1047, Suntec, Singapore, August. Association for Computational Linguistics.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research (JMLR)*, 2:45–66.

David Vickrey, Oscar Kipersztok, and Daphne Koller. 2010. An active learning approach to finding related terms. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, July. Association for Computational Linguistics.

Andreas Vlachos. 2008. A stopping criterion for active learning. *Computer Speech and Language*, 22(3):295–312.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the NAACL-2006 Workshop on Statistical Machine Translation (WMT06)*, New York, New York.

Creating Robust Supervised Classifiers via Web-Scale N-gram Data

Shane Bergsma
University of Alberta
sbergsma@ualberta.ca

Emily Pitler
University of Pennsylvania
epitler@seas.upenn.edu

Dekang Lin
Google, Inc.
lindek@google.com

Abstract

In this paper, we systematically assess the value of using web-scale N-gram data in state-of-the-art supervised NLP classifiers. We compare classifiers that include or exclude features for the counts of various N-grams, where the counts are obtained from a web-scale auxiliary corpus. We show that including N-gram count features can advance the state-of-the-art accuracy on standard data sets for adjective ordering, spelling correction, noun compound bracketing, and verb part-of-speech disambiguation. More importantly, when operating on new domains, or when labeled training data is not plentiful, we show that using web-scale N-gram features is essential for achieving robust performance.

1 Introduction

Many NLP systems use web-scale N-gram counts (Keller and Lapata, 2003; Nakov and Hearst, 2005; Brants et al., 2007). Lapata and Keller (2005) demonstrate good performance on eight tasks using unsupervised web-based models. They show web counts are superior to counts from a large corpus. Bergsma et al. (2009) propose unsupervised and supervised systems that use counts from Google's N-gram corpus (Brants and Franz, 2006). Web-based models perform particularly well on *generation* tasks, where systems choose between competing sequences of output text (such as different spellings), as opposed to *analysis* tasks, where systems choose between abstract labels (such as part-of-speech tags or parse trees).

In this work, we address two natural and related questions which these previous studies leave open:

1. Is there a benefit in combining web-scale counts with the features used in state-of-the-art supervised approaches?
2. How well do web-based models perform on new domains or when labeled data is scarce?

We address these questions on two generation and two analysis tasks, using both existing N-gram data and a novel web-scale N-gram corpus that includes part-of-speech information (Section 2). While previous work has combined web-scale features with other features in specific classification problems (Modjeska et al., 2003; Yang et al., 2005; Vadas and Curran, 2007b), we provide a multi-task, multi-domain comparison.

Some may question why supervised approaches are needed at all for generation problems. Why not solely rely on direct evidence from a giant corpus? For example, for the task of prenominal adjective ordering (Section 3), a system that needs to describe a ball that is both big and red can simply check that *big red* is more common on the web than *red big*, and order the adjectives accordingly.

It is, however, suboptimal to only use N-gram data. For example, ordering adjectives by direct web evidence performs 7% worse than our best supervised system (Section 3.2). No matter how large the web becomes, there will always be plausible constructions that never occur. For example, there are currently no pages indexed by Google with the preferred adjective ordering for *bedraggled 56-year-old [professor]*. Also, in a particular domain, words may have a non-standard usage. Systems trained on labeled data can learn the domain usage and leverage other regularities, such as suffixes and transitivity for adjective ordering.

With these benefits, systems trained on labeled data have become the dominant technology in academic NLP. There is a growing recognition, however, that these systems are highly domain dependent. For example, parsers trained on annotated newspaper text perform poorly on other genres (Gildea, 2001). While many approaches have adapted NLP systems to specific domains (Tsuruoka et al., 2005; McClosky et al., 2006; Blitzer

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 865–874,
Uppsala, Sweden, 11-16 July 2010.

et al., 2007; Daumé III, 2007; Rimell and Clark, 2008), these techniques assume the system knows on which domain it is being used, and that it has access to representative data in that domain. These assumptions are unrealistic in many real-world situations; for example, when automatically processing a heterogeneous collection of web pages. How well do supervised and unsupervised NLP systems perform when used uncustomized, *out-of-the-box* on new domains, and how can we best design our systems for robust *open-domain* performance?

Our results show that using web-scale N-gram data in supervised systems advances the state-of-the-art performance on standard analysis and generation tasks. More importantly, when operating out-of-domain, or when labeled data is not plentiful, using web-scale N-gram data not only helps achieve good performance – it is essential.

2 Experiments and Data

2.1 Experimental Design

We evaluate the benefit of N-gram data on multi-class classification problems. For each task, we have some labeled data indicating the correct output for each example. We evaluate with **accuracy**: the percentage of examples correctly classified in test data. We use one *in-domain* and two *out-of-domain* test sets for each task. Statistical significance is assessed with McNemar's test, $p<0.01$.

We provide results for unsupervised approaches and the majority-class baseline for each task.

For our supervised approaches, we represent the examples as feature vectors, and learn a classifier on the training vectors. There are two feature classes: features that use N-grams (N-GM) and those that do not (LEX). N-GM features are real-valued features giving the log-count of a particular N-gram in the auxiliary web corpus. LEX features are binary features that indicate the presence or absence of a particular string at a given position in the input. The name LEX emphasizes that they identify specific lexical items. The instantiations of both types of features depend on the task and are described in the corresponding sections.

Each classifier is a linear Support Vector Machine (SVM), trained using LIBLINEAR (Fan et al., 2008) on the standard domain. We use the one-vs-all strategy when there are more than two classes (in Section 4). We plot learning curves to measure the accuracy of the classifier when the number of labeled training examples varies. The size of the N-gram data and its counts remain constant. We always optimize the SVM's (L2) regularization parameter on the in-domain development set. We present results with L2-SVM, but achieve similar results with L1-SVM and logistic regression.

2.2 Tasks and Labeled Data

We study two generation tasks: prenominal adjective ordering (Section 3) and context-sensitive spelling correction (Section 4), followed by two analysis tasks: noun compound bracketing (Section 5) and verb part-of-speech disambiguation (Section 6). In each section, we provide references to the origin of the labeled data. For the out-of-domain Gutenberg and Medline data used in Sections 3 and 4, we generate examples ourselves.[1] We chose Gutenberg and Medline in order to provide challenging, distinct domains from our training corpora. Our Gutenberg corpus consists of out-of-copyright books, automatically downloaded from the Project Gutenberg website.[2] The Medline data consists of a large collection of online biomedical abstracts. We describe how labeled adjective and spelling examples are created from these corpora in the corresponding sections.

2.3 Web-Scale Auxiliary Data

The most widely-used N-gram corpus is the Google 5-gram Corpus (Brants and Franz, 2006).

For our tasks, we also use **Google V2**: a new N-gram corpus (also with N-grams of length one-to-five) that we created from the same one-trillion-word snapshot of the web as the Google 5-gram Corpus, but with several enhancements. These include: 1) Reducing noise by removing duplicate sentences and sentences with a high proportion of non-alphanumeric characters (together filtering about 80% of the source data), 2) pre-converting all digits to the 0 character to reduce sparsity for numeric expressions, and 3) including the part-of-speech (POS) tag distribution for each N-gram. The source data was automatically tagged with TnT (Brants, 2000), using the Penn Treebank tag set. Lin et al. (2010) provide more details on the

[1] `http://webdocs.cs.ualberta.ca/~bergsma/Robust/` provides our Gutenberg corpus, a link to Medline, and also the generated examples for both Gutenberg and Medline.

[2] `www.gutenberg.org`. All books just released in 2009 and thus unlikely to occur in the source data for our N-gram corpus (from 2006). Of course, with removal of sentence duplicates and also N-gram thresholding, the possible presence of a test sentence in the massive source data is unlikely to affect results. Carlson et al. (2008) reach a similar conclusion.

N-gram data and N-gram search tools.

The third enhancement is especially relevant here, as we can use the POS distribution to collect counts for N-grams of mixed words and tags. For example, we have developed an N-gram search engine that can count how often the adjective *unprecedented* precedes another adjective in our web corpus (113K times) and how often it follows one (11K times). Thus, even if we haven't seen a particular adjective pair directly, we can use the positional preferences of each adjective to order them.

Early web-based models used search engines to collect N-gram counts, and thus could not use capitalization, punctuation, and annotations such as part-of-speech (Kilgarriff and Grefenstette, 2003). Using a POS-tagged web corpus goes a long way to addressing earlier criticisms of web-based NLP.

3 Prenominal Adjective Ordering

Prenominal adjective ordering strongly affects text readability. For example, while *the unprecedented statistical revolution* is fluent, *the statistical unprecedented revolution* is not. Many NLP systems need to handle adjective ordering robustly. In machine translation, if a noun has two adjective modifiers, they must be ordered correctly in the target language. Adjective ordering is also needed in Natural Language Generation systems that produce information from databases; for example, to convey information (in sentences) about medical patients (Shaw and Hatzivassiloglou, 1999).

We focus on the task of ordering a pair of adjectives independently of the noun they modify and achieve good performance in this setting. Following the set-up of Malouf (2000), we experiment on the 263K adjective pairs Malouf extracted from the British National Corpus (BNC). We use 90% of pairs for training, 5% for testing, and 5% for development. This forms our in-domain data.[3]

We create out-of-domain examples by tokenizing Medline and Gutenberg (Section 2.2), then POS-tagging them with CRFTagger (Phan, 2006). We create examples from all sequences of two adjectives followed by a noun. Like Malouf (2000), we assume that edited text has adjectives ordered fluently. We extract 13K and 9.1K out-of-domain pairs from Gutenberg and Medline, respectively.[4]

[3]BNC is not a domain *per se* (rather a balanced corpus), but has a style and vocabulary distinct from our OOD data.

[4]Like Malouf (2000), we convert our pairs to lower-case. Since the N-gram data includes case, we merge counts from the upper and lower case combinations.

The input to the system is a pair of adjectives, (a_1, a_2), ordered alphabetically. The task is to classify this order as correct (the positive class) or incorrect (the negative class). Since both classes are equally likely, the majority-class **baseline** is around 50% on each of the three test sets.

3.1 Supervised Adjective Ordering

3.1.1 LEX features

Our adjective ordering model with LEX features is a novel contribution of this paper.

We begin with two features for each pair: an indicator feature for a_1, which gets a feature value of $+1$, and an indicator feature for a_2, which gets a feature value of -1. The parameters of the model are therefore weights on specific adjectives. The higher the weight on an adjective, the more it is preferred in the first position of a pair. If the alphabetic ordering is correct, the weight on a_1 should be higher than the weight on a_2, so that the classifier returns a positive score. If the reverse ordering is preferred, a_2 should receive a higher weight. Training the model in this setting is a matter of assigning weights to all the observed adjectives such that the training pairs are maximally ordered correctly. The feature weights thus implicitly produce a linear ordering of all observed adjectives. The examples can also be regarded as rank constraints in a discriminative ranker (Joachims, 2002). Transitivity is achieved naturally in that if we correctly order pairs $a \prec b$ and $b \prec c$ in the training set, then $a \prec c$ by virtue of the weights on a and c.

While exploiting transitivity has been shown to improve adjective ordering, there are many conflicting pairs that make a strict linear ordering of adjectives impossible (Malouf, 2000). We therefore provide an indicator feature for the pair $a_1 a_2$, so the classifier can memorize exceptions to the linear ordering, breaking strict order transitivity. Our classifier thus operates along the lines of rankers in the *preference-based setting* as described in Ailon and Mohri (2008).

Finally, we also have features for all suffixes of length 1-to-4 letters, as these encode useful information about adjective class (Malouf, 2000). Like the adjective features, the suffix features receive a value of $+1$ for adjectives in the first position and -1 for those in the second.

3.1.2 N-GM features

Lapata and Keller (2005) propose a web-based approach to adjective ordering: take the most-

System	IN	O1	O2
Malouf (2000)	91.5	65.6	71.6
web $c(a_1, a_2)$ vs. $c(a_2, a_1)$	87.1	83.7	86.0
SVM with N-GM features	90.0	**85.8**	**88.5**
SVM with LEX features	93.0	70.0	73.9
SVM with N-GM + LEX	**93.7**	83.6	85.4

Table 1: Adjective ordering accuracy (%). SVM and Malouf (2000) trained on BNC, tested on BNC (IN), Gutenberg (O1), and Medline (O2).

frequent order of the words on the web, $c(a_1, a_2)$ vs. $c(a_2, a_1)$. We adopt this as our unsupervised approach. We merge the counts for the adjectives occurring contiguously and separated by a comma.

These are indubitably the most important N-GM features; we include them but also other, tag-based counts from Google V2. Raw counts include cases where one of the adjectives is not used as a modifier: "the *special present* was" vs. "the *present special* issue." We include log-counts for the following, more-targeted patterns:[5] $c(a_1\ a_2$ N.*), $c(a_2\ a_1$ N.*), c(DT $a_1\ a_2$ N.*), c(DT $a_2\ a_1$ N.*). We also include features for the log-counts of each adjective preceded or followed by a word matching an adjective-tag: $c(a_1$ J.*), c(J.* a_1), $c(a_2$ J.*), c(J.* a_2). These assess the positional preferences of each adjective. Finally, we include the log-frequency of each adjective. The more frequent adjective occurs first 57% of the time.

As in all tasks, the counts are features in a classifier, so the importance of the different patterns is weighted discriminatively during training.

3.2 Adjective Ordering Results

In-domain, with both feature classes, we set a strong new standard on this data: 93.7% accuracy for the N-GM+LEX system (Table 1). We trained and tested Malouf (2000)'s program on our data; our LEX classifier, which also uses no auxiliary corpus, makes 18% fewer errors than Malouf's system. Our web-based N-GM model is also superior to the direct evidence web-based approach of Lapata and Keller (2005), scoring 90.0% vs. 87.1% accuracy. These results show the benefit of our new lexicalized and web-based features.

Figure 1 gives the in-domain learning curve. With fewer training examples, the systems with N-GM features strongly outperform the LEX-only system. Note that with tens of thousands of test examples, all differences are highly significant.

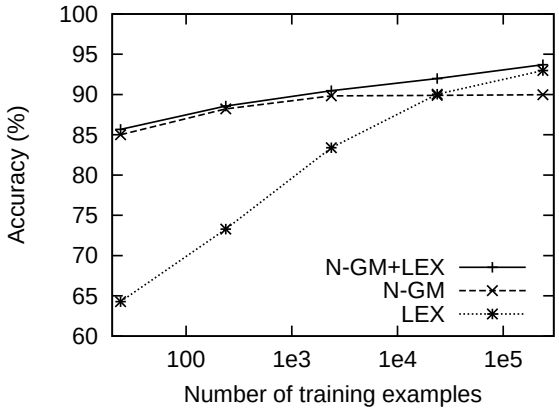


Figure 1: In-domain learning curve of adjective ordering classifiers on BNC.

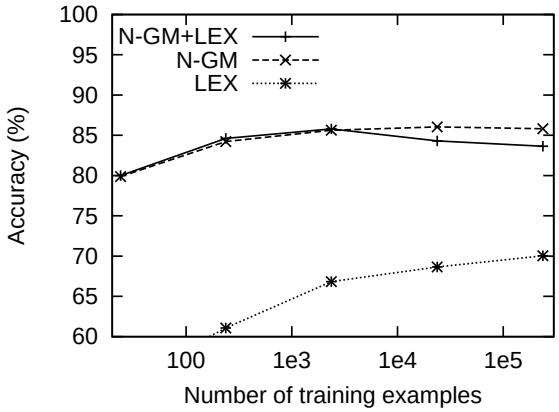


Figure 2: Out-of-domain learning curve of adjective ordering classifiers on Gutenberg.

Out-of-domain, LEX's accuracy drops a shocking 23% on Gutenberg and 19% on Medline (Table 1). Malouf (2000)'s system fares even worse. The overlap between training and test pairs helps explain. While 59% of the BNC test pairs were seen in the training corpus, only 25% of Gutenberg and 18% of Medline pairs were seen in training.

While other ordering models have also achieved "very poor results" out-of-domain (Mitchell, 2009), we expected our expanded set of LEX features to provide good generalization on new data. Instead, LEX is very unreliable on new domains.

N-GM features do not rely on specific pairs in training data, and thus remain fairly robust cross-domain. Across the three test sets, 84-89% of examples had the correct ordering appear at least once on the web. On new domains, the learned N-GM system maintains an advantage over the unsupervised $c(a_1, a_2)$ vs. $c(a_2, a_1)$, but the difference is reduced. Note that training with 10-fold

[5] In this notation, capital letters (and regular expressions) are matched against tags while a_1 and a_2 match words.

cross validation, the N-GM system can achieve up to 87.5% on Gutenberg (90.0% for N-GM + LEX).

The learning curve showing performance on Gutenberg (but still training on BNC) is particularly instructive (Figure 2, performance on Medline is very similar). The LEX system performs much worse than the web-based models across all training sizes. For our top in-domain system, N-GM + LEX, as you add more labeled examples, performance begins *decreasing* out-of-domain. The system disregards the robust N-gram counts as it is more and more confident in the LEX features, and it suffers the consequences.

4 Context-Sensitive Spelling Correction

We now turn to the generation problem of context-sensitive spelling correction. For every occurrence of a word in a pre-defined set of confusable words (like *peace* and *piece*), the system must select the most likely word from the set, flagging possible usage errors when the predicted word disagrees with the original. Contextual spell checkers are one of the most widely used NLP technologies, reaching millions of users via compressed N-gram models in Microsoft Office (Church et al., 2007).

Our in-domain examples are from the New York Times (NYT) portion of Gigaword, from Bergsma et al. (2009). They include the 5 confusion sets where accuracy was below 90% in Golding and Roth (1999). There are 100K training, 10K development, and 10K test examples for each confusion set. Our results are averages across confusion sets.

Out-of-domain examples are again drawn from Gutenberg and Medline. We extract all instances of words that are in one of our confusion sets, along with surrounding context. By assuming the extracted instances represent correct usage, we label 7.8K and 56K out-of-domain test examples for Gutenberg and Medline, respectively.

We test three unsupervised systems: 1) Lapata and Keller (2005) use one token of context on the left and one on the right, and output the candidate from the confusion set that occurs most frequently in this pattern. 2) Bergsma et al. (2009) measure the frequency of the candidates in all the 3-to-5-gram patterns that span the confusable word. For each candidate, they sum the log-counts of all patterns filled with the candidate, and output the candidate with the highest total. 3) The **baseline** predicts the most frequent member of each confusion set, based on frequencies in the NYT training data.

System	IN	O1	O2
Baseline	66.9	44.6	60.6
Lapata and Keller (2005)	88.4	78.0	87.4
Bergsma et al. (2009)	94.8	87.7	94.2
SVM with N-GM features	95.7	**92.1**	93.9
SVM with LEX features	95.2	85.8	91.0
SVM with N-GM + LEX	**96.5**	91.9	**94.8**

Table 2: Spelling correction accuracy (%). SVM trained on NYT, tested on NYT (IN) and out-of-domain Gutenberg (O1) and Medline (O2).

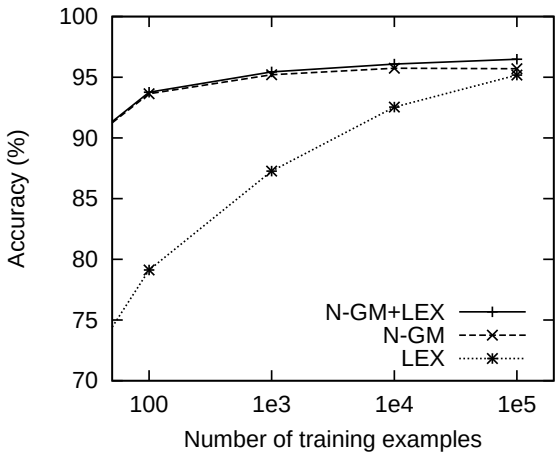


Figure 3: In-domain learning curve of spelling correction classifiers on NYT.

4.1 Supervised Spelling Correction

Our LEX features are typical disambiguation features that flag specific aspects of the context. We have features for the words at all positions in a 9-word window (called collocation features by Golding and Roth (1999)), plus indicators for a particular word preceding or following the confusable word. We also include indicators for all N-grams, and their position, in a 9-word window.

For N-GM count features, we follow Bergsma et al. (2009). We include the log-counts of all N-grams that span the confusable word, with each word in the confusion set filling the N-gram pattern. These features do not use part-of-speech. Following Bergsma et al. (2009), we get N-gram counts using the original Google N-gram Corpus.

While neither our LEX nor N-GM features are novel on their own, they have, perhaps surprisingly, not yet been evaluated in a single model.

4.2 Spelling Correction Results

The N-GM features outperform the LEX features, 95.7% vs. 95.2% (Table 2). Together, they achieve a very strong 96.5% in-domain accuracy.

This is 2% higher than the best unsupervised approach (Bergsma et al., 2009). Web-based models again perform well across a range of training data sizes (Figure 3).

The error rate of LEX nearly triples on Gutenberg and almost doubles on Medline (Table 2). Removing N-GM features from the N-GM + LEX system, errors increase around 75% on both Gutenberg and Medline. The LEX features provide no help to the combined system on Gutenberg, while they do help significantly on Medline. Note the learning curves for N-GM+LEX on Gutenberg and Medline (not shown) do not display the decrease that we observed in adjective ordering (Figure 2).

Both the baseline and LEX perform poorly on Gutenberg. The baseline predicts the majority class from NYT, but it's not always the majority class in Gutenberg. For example, while in NYT *site* occurs 87% of the time for the *(cite, sight, site)* confusion set, *sight* occurs 90% of the time in Gutenberg. The LEX classifier exploits this bias as it is regularized toward a more economical model, but the bias does not transfer to the new domain.

5 Noun Compound Bracketing

About 70% of web queries are noun phrases (Barr et al., 2008) and methods that can reliably parse these phrases are of great interest in NLP. For example, a web query for *zebra hair straightener* should be bracketed as *(zebra (hair straightener))*, a stylish hair straightener with zebra print, rather than *((zebra hair) straightener)*, a useless product since the fur of zebras is already quite straight.

The noun compound (NC) bracketing task is usually cast as a decision whether a 3-word NC has a left or right bracketing. Most approaches are unsupervised, using a large corpus to compare the statistical association between word pairs in the NC. The adjacency model (Marcus, 1980) proposes a left bracketing if the association between words one and two is higher than between two and three. The dependency model (Lauer, 1995a) compares one-two vs. *one-three*. We include dependency model results using PMI as the association measure; results were lower with the adjacency model.

As in-domain data, we use Vadas and Curran (2007a)'s Wall-Street Journal (WSJ) data, an extension of the Treebank (which originally left NPs flat). We extract all sequences of three consecutive common nouns, generating 1983 examples from sections 0-22 of the Treebank as training, 72 from section 24 for development and 95 from section 23 as a test set. As out-of-domain data, we use 244 NCs from Grolier Encyclopedia (Lauer, 1995a) and 429 NCs from Medline (Nakov, 2007).

The majority class **baseline** is left-bracketing.

System	IN	O1	O2
Baseline	70.5	66.8	84.1
Dependency model	74.7	**82.8**	84.4
SVM with N-GM features	89.5	81.6	86.2
SVM with LEX features	81.1	70.9	79.0
SVM with N-GM + LEX	**91.6**	81.6	**87.4**

Table 3: NC-bracketing accuracy (%). SVM trained on WSJ, tested on WSJ (IN) and out-of-domain Grolier (O1) and Medline (O2).

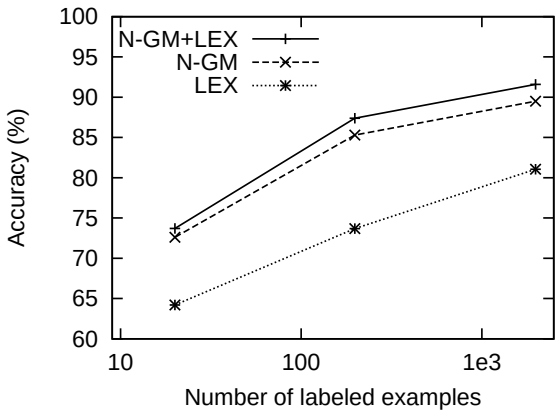


Figure 4: In-domain NC-bracketer learning curve

5.1 Supervised Noun Bracketing

Our LEX features indicate the specific noun at each position in the compound, plus the three pairs of nouns and the full noun triple. We also add features for the capitalization pattern of the sequence.

N-GM features give the log-count of all subsets of the compound. Counts are from Google V2. Following Nakov and Hearst (2005), we also include counts of noun pairs collapsed into a single token; if a pair occurs often on the web as a single unit, it strongly indicates the pair is a constituent.

Vadas and Curran (2007a) use simpler features, e.g. they do not use collapsed pair counts. They achieve 89.9% in-domain on WSJ and 80.7% on Grolier. Vadas and Curran (2007b) use comparable features to ours, but do not test out-of-domain.

5.2 Noun Compound Bracketing Results

N-GM systems perform much better on this task (Table 3). N-GM+LEX is statistically significantly

better than LEX on all sets. In-domain, errors more than double without N-GM features. LEX performs poorly here because there are far fewer training examples. The learning curve (Figure 4) looks much like earlier in-domain curves (Figures 1 and 3), but truncated before LEX becomes competitive. The absence of a sufficient amount of labeled data explains why NC-bracketing is generally regarded as a task where corpus counts are crucial.

All web-based models (including the dependency model) exceed 81.5% on Grolier, which is the level of human agreement (Lauer, 1995b). N-GM + LEX is highest on Medline, and close to the 88% human agreement (Nakov and Hearst, 2005). Out-of-domain, the LEX approach performs very poorly, close to or below the baseline accuracy. With little training data and cross-domain usage, N-gram features are essential.

6 Verb Part-of-Speech Disambiguation

Our final task is POS-tagging. We focus on one frequent and difficult tagging decision: the distinction between a past-tense verb (VBD) and a past participle (VBN). For example, in *the troops stationed in Iraq*, the verb *stationed* is a VBN; *troops* is the head of the phrase. On the other hand, for *the troops vacationed in Iraq*, the verb *vacationed* is a VBD and also the head. Some verbs make the distinction explicit (*eat* has VBD *ate*, VBN *eaten*), but most require context for resolution.

Conflating VBN/VBD is damaging because it affects downstream parsers and semantic role labelers. The task is difficult because nearby POS tags can be identical in both cases. When the verb follows a noun, tag assignment can hinge on world-knowledge, i.e., the global lexical relation between the noun and verb (E.g., *troops* tends to be the object of *stationed* but the subject of *vacationed*).[6] Web-scale N-gram data might help improve the VBN/VBD distinction by providing relational evidence, even if the verb, noun, or verb-noun pair were not observed in training data.

We extract nouns followed by a VBN/VBD in the WSJ portion of the Treebank (Marcus et al., 1993), getting 23K training, 1091 development and 1130 test examples from sections 2-22, 24, and 23, respectively. For out-of-domain data, we get 21K examples from the Brown portion of the Treebank and 6296 examples from tagged Medline abstracts in the PennBioIE corpus (Kulick et al., 2004).

The majority class **baseline** is to choose VBD.

6.1 Supervised Verb Disambiguation

There are two orthogonal sources of information for predicting VBN/VBD: 1) the noun-verb pair, and 2) the context around the pair. Both N-GM and LEX features encode both these sources.

6.1.1 LEX features

For 1), we use indicators for the noun and verb, the noun-verb pair, whether the verb is on an in-house list of *said*-verb (like *warned*, *announced*, etc.), whether the noun is capitalized and whether it's upper-case. Note that in training data, 97.3% of capitalized nouns are followed by a VBD and 98.5% of *said*-verbs are VBDs. For 2), we provide indicator features for the words before the noun and after the verb.

6.1.2 N-GM features

For 1), we characterize a noun-verb relation via features for the pair's distribution in Google V2. Characterizing a word by its distribution has a long history in NLP; we apply similar techniques to *relations*, like Turney (2006), but with a larger corpus and richer annotations. We extract the 20 most-frequent N-grams that contain both the noun and the verb in the pair. For each of these, we convert the tokens to POS-tags, except for tokens that are among the most frequent 100 unigrams in our corpus, which we include in word form. We mask the noun of interest as *N* and the verb of interest as *V*. This converted N-gram is the feature label. The value is the pattern's log-count. A high count for patterns like *(N that V)*, *(N have V)* suggests the relation is a VBD, while patterns *(N that were V)*, *(N V by)*, *(V some N)* indicate a VBN. As always, the classifier learns the association between patterns and classes.

For 2), we use counts for the verb's context co-occurring with a VBD or VBN tag. E.g., we see whether VBD cases like *troops ate* or VBN cases like *troops eaten* are more frequent. Although our corpus contains many VBN/VBD errors, we hope the errors are random enough for aggregate counts to be useful. The context is an N-gram spanning the VBN/VBD. We have log-count features for all five such N-grams in the (previous-word, noun, verb, next-word) quadruple. The log-count is in-

[6] HMM-style taggers, like the fast TnT tagger used on our web corpus, do not use bilexical features, and so perform especially poorly on these cases. One motivation for our work was to develop a fast post-processor to fix VBN/VBD errors.

System	IN	O1	O2
Baseline	89.2	85.2	79.6
ContextSum	92.5	91.1	90.4
SVM with N-GM features	96.1	93.4	93.8
SVM with LEX features	95.8	93.4	93.0
SVM with N-GM + LEX	**96.4**	**93.5**	**94.0**

Table 4: Verb-POS-disambiguation accuracy (%) trained on WSJ, tested on WSJ (IN) and out-of-domain Brown (O1) and Medline (O2).

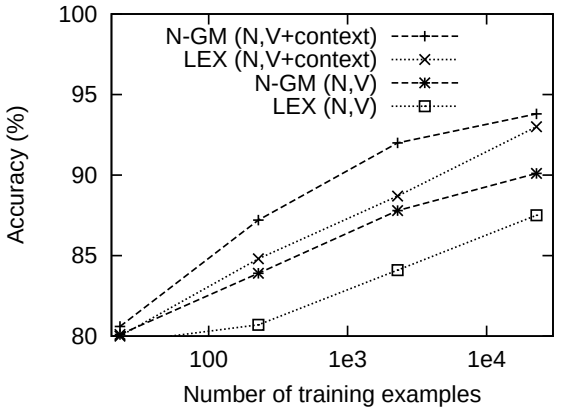


Figure 5: Out-of-domain learning curve of verb disambiguation classifiers on Medline.

dexed by the position and length of the N-gram. We include separate count features for contexts matching the specific noun and for when the noun token can match any word tagged as a noun.

ContextSum: We use these context counts in an unsupervised system, ContextSum. Analogously to Bergsma et al. (2009), we separately sum the log-counts for all contexts filled with VBD and then VBN, outputting the tag with the higher total.

6.2 Verb POS Disambiguation Results

As in all tasks, N-GM+LEX has the best in-domain accuracy (96.4%, Table 4). Out-of-domain, when N-grams are excluded, errors only increase around 14% on Medline and 2% on Brown (the differences are not statistically significant). Why? Figure 5, the learning curve for performance on Medline, suggests some reasons. We omit N-GM+LEX from Figure 5 as it closely follows N-GM.

Recall that we grouped the features into two views: 1) noun-verb (N,V) and 2) context. If we use just (N,V) features, we do see a large drop out-of-domain: LEX (N,V) lags N-GM (N,V) even using all the training examples. The same is true using only context features (not shown). Using both views, the results are closer: 93.8% for N-GM and 93.0% for LEX. With two views of an example, LEX is more likely to have domain-neutral features to draw on. Data sparsity is reduced.

Also, the Treebank provides an atypical number of labeled examples for analysis tasks. In a more typical situation with less labeled examples, N-GM strongly dominates LEX, even when two views are used. E.g., with 2285 training examples, N-GM+LEX is statistically significantly better than LEX on both out-of-domain sets.

All systems, however, perform log-linearly with training size. In other tasks we only had a handful of N-GM features; here there are 21K features for the distributional patterns of N,V pairs. Reducing this feature space by pruning or performing transformations may improve accuracy in and out-of-domain.

7 Discussion and Future Work

Of all classifiers, LEX performs worst on all cross-domain tasks. Clearly, many of the regularities that a typical classifier exploits in one domain do not transfer to new genres. N-GM features, however, do not depend directly on training examples, and thus work better cross-domain. Of course, using web-scale N-grams is not the only way to create robust classifiers. Counts from any large auxiliary corpus may also help, but web counts should help more (Lapata and Keller, 2005). Section 6.2 suggests that another way to mitigate domain-dependence is having multiple feature views.

Banko and Brill (2001) argue "a logical next step for the research community would be to direct efforts towards increasing the size of annotated training collections." Assuming we really do want systems that operate beyond the specific domains on which they are trained, the community also needs to identify which systems behave as in Figure 2, where the accuracy of the best in-domain system actually decreases with more training examples. Our results suggest better features, such as web pattern counts, may help more than expanding training data. Also, systems using web-scale unlabeled data will improve automatically as the web expands, without annotation effort.

In some sense, using web counts as features is a form of domain adaptation: adapting a web model to the training domain. How do we ensure these features are adapted well and not used in domain-specific ways (especially with many features to adapt, as in Section 6)? One option may

be to regularize the classifier specifically for out-of-domain accuracy. We found that adjusting the SVM misclassification penalty (for more regularization) can help or hurt out-of-domain. Other regularizations are possible. In each task, there are domain-neutral unsupervised approaches. We could encode these systems as linear classifiers with corresponding weights. Rather than a typical SVM that minimizes the weight-norm $||w||$ (plus the slacks), we could regularize toward domain-neutral weights. This regularization could be optimized on creative splits of the training data.

8 Conclusion

We presented results on tasks spanning a range of NLP research: generation, disambiguation, parsing and tagging. Using web-scale N-gram data improves accuracy on each task. When less training data is used, or when the system is used on a different domain, N-gram features greatly improve performance. Since most supervised NLP systems do not use web-scale counts, further cross-domain evaluation may reveal some very brittle systems. Continued effort in new domains should be a priority for the community going forward.

Acknowledgments

We gratefully acknowledge the Center for Language and Speech Processing at Johns Hopkins University for hosting the workshop at which part of this research was conducted.

References

Nir Ailon and Mehryar Mohri. 2008. An efficient reduction of ranking to classification. In *COLT*.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *ACL*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2009. Web-scale N-gram models for lexical disambiguation. In *IJCAI*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP*.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *ANLP*.

Andrew Carlson, Tom M. Mitchell, and Ian Fette. 2008. Data analysis project: Leveraging massive textual corpora using n-gram statistics. Technial Report CMU-ML-08-107.

Kenneth Church, Ted Hart, and Jianfeng Gao. 2007. Compressing trigram language models with Golomb coding. In *EMNLP-CoNLL*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9.

Dan Gildea. 2001. Corpus variation and parser performance. In *EMNLP*.

Andrew R. Golding and Dan Roth. 1999. A Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the Web as corpus. *Computational Linguistics*, 29(3):333–347.

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *BioLINK 2004: Linking Biological Literature, Ontologies and Databases*.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

Mark Lauer. 1995a. Corpus statistics meet the noun compound: Some empirical results. In *ACL*.

Mark Lauer. 1995b. *Designing Statistical Language Learners: Experiments on Compound Nouns*. Ph.D. thesis, Macquarie University.

Dekang Lin, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale. 2010. New tools for web-scale N-grams. In *LREC*.

Robert Malouf. 2000. The order of prenominal adjectives in natural language generation. In *ACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mitchell P. Marcus. 1980. *Theory of Syntactic Recognition for Natural Languages*. MIT Press, Cambridge, MA, USA.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *COLING-ACL*.

Margaret Mitchell. 2009. Class-based ordering of prenominal modifiers. In *12th European Workshop on Natural Language Generation*.

Natalia N. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the Web in machine learning for *other*-anaphora resolution. In *EMNLP*.

Preslav Nakov and Marti Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*.

Preslav Ivanov Nakov. 2007. *Using the Web as an Implicit Training Set: Application to Noun Compound Syntax and Semantics*. Ph.D. thesis, University of California, Berkeley.

Xuan-Hieu Phan. 2006. CRFTagger: CRF English POS Tagger. `crftagger.sourceforge.net`.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *EMNLP*.

James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among premodifiers. In *ACL*.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Advances in Informatics*.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

David Vadas and James R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *ACL*.

David Vadas and James R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *PACLING*.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *ACL*.

Convolution Kernel over Packed Parse Forest

Min Zhang Hui Zhang Haizhou Li
Institute for Infocomm Research
A-STAR, Singapore
{mzhang,vishz,hli}@i2r.a-star.edu.sg

Abstract

This paper proposes a convolution forest kernel to effectively explore rich structured features embedded in a packed parse forest. As opposed to the convolution tree kernel, the proposed forest kernel does not have to commit to a single best parse tree, is thus able to explore very large object spaces and much more structured features embedded in a forest. This makes the proposed kernel more robust against parsing errors and data sparseness issues than the convolution tree kernel. The paper presents the formal definition of convolution forest kernel and also illustrates the computing algorithm to fast compute the proposed convolution forest kernel. Experimental results on two NLP applications, relation extraction and semantic role labeling, show that the proposed forest kernel significantly outperforms the baseline of the convolution tree kernel.

1 Introduction

Parse tree and packed forest of parse trees are two widely used data structures to represent the syntactic structure information of sentences in natural language processing (NLP). The structured features embedded in a parse tree have been well explored together with different machine learning algorithms and proven very useful in many NLP applications (Collins and Duffy, 2002; Moschitti, 2004; Zhang et al., 2007). A forest (Tomita, 1987) compactly encodes an exponential number of parse trees. In this paper, we study how to effectively explore structured features embedded in a forest using convolution kernel (Haussler, 1999).

As we know, feature-based machine learning methods are less effective in modeling highly structured objects (Vapnik, 1998), such as parse tree or semantic graph in NLP. This is due to the fact that it is usually very hard to represent structured objects using vectors of reasonable dimensions without losing too much information. For example, it is computationally infeasible to enumerate all subtree features (using subtree a feature) for a parse tree into a linear feature vector. Kernel-based machine learning method is a good way to overcome this problem. Kernel methods employ a kernel function, that must satisfy the properties of being symmetric and positive, to measure the similarity between two objects by computing implicitly the dot product of certain features of the input objects in high (or even infinite) dimensional feature spaces without enumerating all the features (Vapnik, 1998).

Many learning algorithms, such as SVM (Vapnik, 1998), the Perceptron learning algorithm (Rosenblatt, 1962) and Voted Perceptron (Freund and Schapire, 1999), can work directly with kernels by replacing the dot product with a particular kernel function. This nice property of kernel methods, that implicitly calculates the dot product in a high-dimensional space over the original representations of objects, has made kernel methods an effective solution to modeling structured objects in NLP.

In the context of parse tree, convolution tree kernel (Collins and Duffy, 2002) defines a feature space consisting of all subtree types of parse trees and counts the number of common subtrees as the syntactic similarity between two parse trees. The tree kernel has shown much success in many NLP applications like parsing (Collins and Duffy, 2002), semantic role labeling (Moschitti, 2004; Zhang et al., 2007), relation extraction (Zhang et al., 2006), pronoun resolution (Yang et al., 2006), question classification (Zhang and Lee, 2003) and machine translation (Zhang and Li, 2009), where the tree kernel is used to compute the similarity between two NLP application instances that are usually represented by parse trees. However, in those studies, the tree kernel only covers the features derived from single 1-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 875–885,
Uppsala, Sweden, 11-16 July 2010.

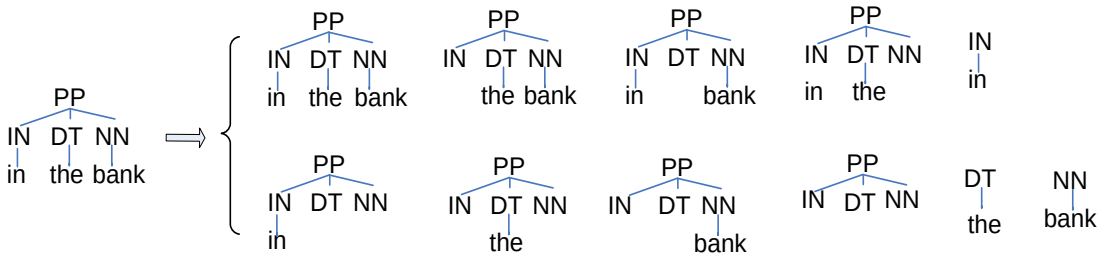


Figure 1. A parse tree and its 11 subtree features covered by convolution tree kernel

best parse tree. This may largely compromise the performance of tree kernel due to parsing errors and data sparseness.

To address the above issues, this paper constructs a forest-based convolution kernel to mine structured features directly from packed forest. A packet forest compactly encodes exponential number of n-best parse trees, and thus containing much more rich structured features than a single parse tree. This advantage enables the forest kernel not only to be more robust against parsing errors, but also to be able to learn more reliable feature values and help to solve the data sparseness issue that exists in the traditional tree kernel. We evaluate the proposed kernel in two real NLP applications, relation extraction and semantic role labeling. Experimental results on the benchmark data show that the forest kernel significantly outperforms the tree kernel.

The rest of the paper is organized as follows. Section 2 reviews the convolution tree kernel while section 3 discusses the proposed forest kernel in details. Experimental results are reported in section 4. Finally, we conclude the paper in section 5.

2 Convolution Kernel over Parse Tree

Convolution kernel was proposed as a concept of kernels for discrete structures by Haussler (1999) and related but independently conceived ideas on string kernels first presented in (Watkins, 1999). The framework defines the kernel function between input objects as the convolution of "sub-kernels", i.e. the kernels for the decompositions (parts) of the input objects.

The parse tree kernel (Collins and Duffy, 2002) is an instantiation of convolution kernel over syntactic parse trees. Given a parse tree, its features defined by a tree kernel are all of its subtree types and the value of a given feature is the number of the occurrences of the subtree in the parse tree. Fig. 1 illustrates a parse tree with all of its 11 subtree features covered by the convolution tree kernel. In the tree kernel, a parse tree T is represented by a vector of integer counts of each *subtree type* (i.e., *subtree* regardless of its ancestors, descendants and span covered):

$$\phi(T) = (\#\ subtreetype_1(T),\ \ldots,\ \#\ subtreetype_n(T))$$

where $\#\ subtreetype_i(T)$ is the occurrence number of the i^{th} subtree type in T. The tree kernel counts the number of common subtrees as the syntactic similarity between two parse trees. Since the number of subtrees is exponential with the tree size, it is computationally infeasible to directly use the feature vector $\phi(T)$. To solve this computational issue, Collins and Duffy (2002) proposed the following tree kernel to calculate the dot product between the above high dimensional vectors implicitly.

$$\begin{aligned} K(T_1,T_2) &= <\phi(T_1),\phi(T_2)> \\ &= \sum_i \#subtreetype_i(T_1)\cdot\#subtreetype_i(T_2) \\ &= \sum_i \left(\sum_{n_1\in N_1} I_{subtree_i}(n_1) \right)\cdot\left(\sum_{n_2\in N_2} I_{subtree_i}(n_2) \right) \\ &= \sum_{n_1\in N_1}\sum_{n_2\in N_2} \Delta(n_1,n_2) \end{aligned}$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively, and $I_{subtree_i}(n)$ is a function that is 1 iff the $subtreetype_i$ occurs with root at node n and zero otherwise, and $\Delta(n_1,n_2)$ is the number of the common *subtrees* rooted at n_1 and n_2, i.e.,

$$\Delta(n_1,n_2) = \sum_i I_{subtree_i}(n_1)\cdot I_{subtree_i}(n_2)$$

$\Delta(n_1,n_2)$ can be computed by the following recursive rules:

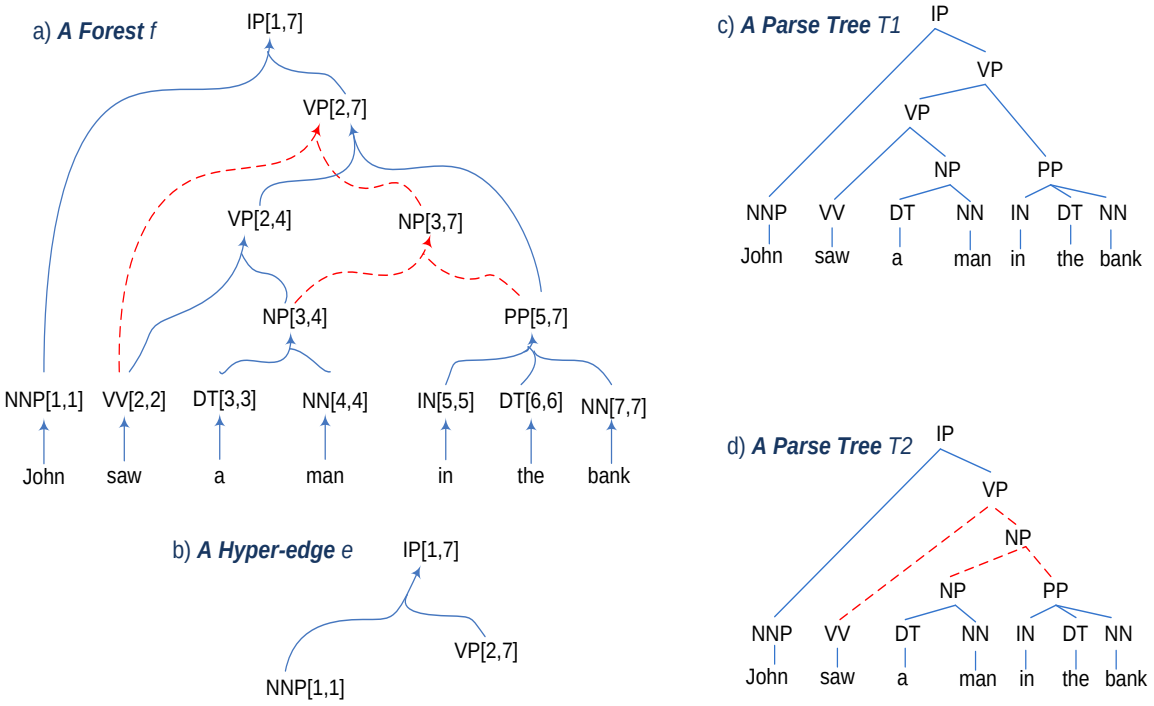


Figure 2. An example of a packed forest, a hyper-edge and two parse trees covered by the packed forest

Rule 1: if the productions (CFG rules) at n_1 and n_2 are different, $\Delta(n_1, n_2) = 0$;

Rule 2: else if both n_1 and n_2 are pre-terminals (POS tags), $\Delta(n_1, n_2) = 1 \times \lambda$;

Rule 3: else,

$$\Delta(n_1, n_2) = \lambda \cdot \prod_{j=1}^{nc(n_1)} (1 + \Delta(ch(n_1, j), ch(n_2, j)))\,,$$

where $nc(n_1)$ is the child number of n_1, $ch(n,j)$ is the j^{th} child of node n and λ ($0 < \lambda \leq 1$) is the decay factor in order to make the kernel value less variable with respect to the *subtree* sizes (Collins and Duffy, 2002). The recursive **Rule 3** holds because given two nodes with the same children, one can construct common subtrees using these children and common subtrees of further offspring. The time complexity for computing this kernel is $O(|N_1| \cdot |N_2|)$.

As discussed in previous section, when convolution tree kernel is applied to NLP applications, its performance is vulnerable to the errors from the single parse tree and data sparseness. In this paper, we present a convolution kernel over packed forest to address the above issues by exploring structured features embedded in a forest.

3 Convolution Kernel over Forest

In this section, we first illustrate the concept of packed forest and then give a detailed discussion on the covered feature space, fractional count, feature value and the forest kernel function itself.

3.1 Packed forest of parse trees

Informally, a packed parse forest, or (packed) forest in short, is a compact representation of all the derivations (i.e. parse trees) for a given sentence under context-free grammar (Tomita, 1987; Billot and Lang, 1989; Klein and Manning, 2001). It is the core data structure used in natural language parsing and other downstream NLP applications, such as syntax-based machine translation (Zhang et al., 2008; Zhang et al., 2009a). In parsing, a sentence corresponds to exponential number of parse trees with different tree probabilities, where a forest can compact all the parse trees by sharing their common subtrees in a bottom-up manner. Formally, a packed forest F can be described as a triple:

$$F = < V, E, S >$$

where V is the set of non-terminal nodes, E is the set of hyper-edges and S is a sentence

represented as an ordered word sequence. A hyper-edge e is a group of edges in a parse tree which connects a father node and its all child nodes, representing a CFG rule. A non-terminal node in a forest is represented as a "label [start, end]", where the "label" is its syntax category and "[start, end]" is the span of words it covers. As shown in Fig. 2, these two parse trees ($T1$ and $T2$) can be represented as a single forest by sharing their common subtrees (such as NP[3,4] and PP[5,7]) and merging common non-terminal nodes covering the same span (such as VP[2,7], where there are two hyper-edges attach to it).

Given the definition of forest, we introduce the concepts of inside probability $\beta(.)$ and outside probability $\alpha(.)$ that are widely-used in parsing (Baker, 1979; Lari and Young, 1990) and are also to be used in our kernel calculation.

$$\beta(v[p,p]) = P(v \to S[p])$$

$$\beta(v[p,q]) = \sum_{\substack{e \text{ is a hyper-edge} \\ \text{attached to } v}} \left(P(e) \cdot \prod_{\substack{c_i[p_i,q_i] \text{is a leaf} \\ \text{node of } e}} \beta(c_i[p_i,q_i]) \right)$$

$$\alpha(root(f)) = 1$$

$$\alpha(v[p,q]) = \sum_{\substack{e \text{ is a hyper-edge and } v \\ \text{is its one leaf node}}} \left(\alpha(root(e)) \cdot P(e) \cdot \prod_{\substack{c_i[p_i,q_i] \text{is a children node} \\ \text{of e except } v}} \beta(c_i[p_i,q_i]) \right)$$

where v is a forest node, $S[p]$ is the p^{th} word of input sentence S, $P(v \to S[p])$ is the probability of the CFG rule $v \to S[p]$, $root(.)$ returns the root node of input structure, $[p_i, q_i]$ is a sub-span of $[p, q]$, being covered by c_i, and $P(e)$ is the PCFG probability of e. From these definitions, we can see that the inside probability is total probability of generating words $S[p,q]$ from non-terminal node $v[p,q]$ while the outside probability is the total probability of generating node $v[p,q]$ and words outside $S[p,q]$ from the root of forest. The inside probability can be calculated using dynamic programming in a bottom-up fashion while the outside probability can be calculated using dynamic programming in a top-to-down way.

3.2 Convolution forest kernel

In this subsection, we first define the feature space covered by forest kernel, and then define the forest kernel function.

3.2.1 Feature space, object space and feature value

The forest kernel counts the number of common subtrees as the syntactic similarity between two forests. Therefore, in the same way as tree kernel, its feature space is also defined as all the possible subtree types that a CFG grammar allows. In a forest kernel, forest *F* is represented by a vector of *fractional counts* of each *subtree type* (subtree regardless of its ancestors, descendants and span covered):

$$\begin{aligned} \phi(F) &= (\#\ subtreetype_1(F), \ldots, \\ &\quad \#\ subtreetype_n(F)) \\ &= (\#subtreetype_1(n\text{-}best\ parse\ trees), \ldots, \\ &\quad \#\ subtreetype_n(n\text{-}best\ parse\ trees)) \end{aligned} \quad (1)$$

where # $subtreetype_i(F)$ is the occurrence number of the i^{th} subtree type ($subtreetype_i$) in forest *F*, i.e., a *n*-best parse tree lists with a huge *n*.

Although the feature spaces of the two kernels are the same, their object spaces (tree *vs.* forest) and feature values (integer counts *vs.* fractional counts) differ very much. A forest encodes exponential number of parse trees, and thus containing exponential times more subtrees than a single parse tree. This ensures forest kernel to learn more reliable feature values and is also able to help to address the data sparseness issues in a better way than tree kernel does. Forest kernel is also expected to yield more non-zero feature values than tree kernel. Furthermore, different parse tree in a forest represents different derivation and interpretation for a given sentence. Therefore, forest kernel should be more robust to parsing errors than tree kernel.

In tree kernel, one occurrence of a subtree contributes 1 to the value of its corresponding feature (*subtree type*), so the feature value is an integer count. However, the case turns out very complicated in forest kernel. In a forest, each of its parse trees, when enumerated, has its own

probability. So one subtree extracted from different parse trees should have different *fractional count* with regard to the probabilities of different parse trees. Following the previous work (Charniak and Johnson, 2005; Huang, 2008), we define the *fractional count* of the occurrence of a *subtree* in a parse tree t_i as

$$c(subtree, t_i) = \begin{cases} 0 & if\ subtree \notin t_i \\ P(subtree, t_i|f,s) & otherwise \end{cases}$$
$$= \begin{cases} 0 & if\ subtree \notin t_i \\ P(t_i|f,s) & otherwise \end{cases}$$

where we have $P(subtree, t_i|f,s) = P(t_i|f,s)$ if $subtree \in t_i$. Then we define the fractional count of the occurrence of a subtree in a forest *f* as

$$\begin{aligned} c(subtree, f) &= P(subtree|f,s) \\ &= \textstyle\sum_{t_i} P(subtree, t_i|f,s) \\ &= \textstyle\sum_{t_i} I_{subtree}\ (t_i) \cdot P(t_i|f,s) \end{aligned} \quad (2)$$

where $I_{subtree}\ (t_i)$ is a binary function that is 1 iif the $subtree \in t_i$ and zero otherwise. Obviously, it needs exponential time to compute the above fractional counts. However, due to the property of forest that compactly represents all the parse trees, the posterior probability of a subtree in a forest, $P(subtree|f,s)$, can be easily computed in an Inside-Outside fashion as the product of three parts: the outside probability of its root node, the probabilities of parse hyperedges involved in the subtree, and the inside probabilities of its leaf nodes (Lari and Young, 1990; Mi and Huang, 2008).

$$\begin{aligned} c(subtree, f) &= P(subtree|f,s) \\ &= \frac{\alpha\beta(subtree)}{\alpha\beta(root(f))} \end{aligned} \quad (3)$$

where

$$\begin{aligned} \alpha\beta(subtree) &= \alpha\big(root(subtree)\big) \\ &\cdot \prod_{e \in subtree} P(e) \\ &\cdot \prod_{v \in leaf\ (subtree\ )} \beta(v) \end{aligned} \quad (4)$$

and

$$\begin{aligned} \alpha\beta\big(root(f)\big) &= \alpha\big(root(f)\big) \cdot \beta\big(root(f)\big) \\ &= \beta\big(root(f)\big) \end{aligned}$$

where $\alpha(.)$ and $\beta(.)$ denote the outside and inside probabilities. They can be easily obtained using the equations introduced at section 3.1.

Given a subtree, we can easily compute its fractional count (i.e. its feature value) directly using eq. (3) and (4) without the need of enumerating each parse trees as shown at eq. (2)[1]. Nonetheless, it is still computationally infeasible to directly use the feature vector $\phi(F)$ (see eq. (1)) by explicitly enumerating all subtrees although its fractional count is easily calculated. In the next subsection, we present the forest kernel that implicitly calculates the dot-product between two $\phi(F)$s in a polynomial time.

3.2.2 Convolution forest kernel

The forest kernel counts the *fractional numbers* of common subtrees as the syntactic similarity between two forests. We define the forest kernel function $K_f(f_1, f_2)$ in the following way.

$$\begin{aligned} K_f(f_1, f_2) &= <\phi(f_1), \phi(f_2)> \\ &= \sum_i \#subtreetype_i(f_1).\#subtreetype_i(f_2) \\ &= \sum_{\substack{subtree\ 1 \in f_1 \\ subtree\ 2 \in f_2}} \Big(I_{eq}(subtree1, subtree2) \\ &\qquad \cdot c(subtree1, f_1) \\ &\qquad \cdot c(subtree2, f_2) \Big) \\ &= \textstyle\sum_{v_1 \in N_1} \sum_{v_2 \in N_2} \Delta'(v_1, v_2) \end{aligned} \quad (5)$$

where

- $I_{eq}(\cdot,\cdot)$ is a binary function that is 1 iif the input two subtrees are identical (i.e. they have the same typology and node labels) and zero otherwise;
- $c(\cdot,\cdot)$ is the fractional count defined at eq. (3);
- N_1 and N_2 are the sets of nodes in forests f_1 and f_2;
- $\Delta'(v_1, v_2)$ returns the accumulated value of products between each two fractional counts of the common subtrees rooted at v_1 and v_2, i.e.,

$$\begin{aligned} &\Delta'(v_1, v_2) \\ &= \sum_{\substack{root\ (subtree\ 1)=v_1 \\ root\ (subtree\ 2)=v_2}} \Big(I_{eq}(subtree1, subtree2) \\ &\qquad \cdot c(subtree1, f_1) \\ &\qquad \cdot c(subtree2, f_2) \Big) \end{aligned}$$

[1] It has been proven in parsing literatures (Baker, 1979; Lari and Young, 1990) that eq. (3) defined by Inside-Outside probabilities is exactly to compute the sum of those parse tree probabilities that cover the subtree of being considered as defined at eq. (2).

We next show that $\Delta'(v_1, v_2)$ can be computed recursively in a polynomial time as illustrated at Algorithm 1. To facilitate discussion, we temporarily ignore all fractional counts in Algorithm 1. Indeed, Algorithm 1 can be viewed as a natural extension of convolution kernel from over tree to over forest. In forest[2], a node can root multiple hyper-edges and each hyper-edge is independent to each other. Therefore, Algorithm 1 iterates each hyper-edge pairs with roots at v_1 and v_2 (line 3-4), and sums over (eq. (7) at line 9) each recursively-accumulated sub-kernel scores of subtree pairs extended from the hyper-edge pair (e_1, e_2) (eq. (6) at line 8). Eq. (7) holds because the hyper-edges attached to the same node are independent to each other. Eq. (6) is very similar to the **Rule 3** of tree kernel (see section 2) except its inputs are hyper-edges and its further expansion is based on forest nodes. Similar to tree kernel (Collins and Duffy, 2002), eq. (6) holds because a common subtree by extending from (e_1, e_2) can be formed by taking the hyper-edge (e_1, e_2), together with a choice at each of their leaf nodes of simply taking the non-terminal at the leaf node, or any one of the common subtrees with root at the leaf node. Thus there are $\left(1 + \Delta'\left(leaf(e_1, j), leaf(e_2, j)\right)\right)$ possible choices at the j^{th} leaf node. In total, there are $\Delta''(e_1, e_2)$ (eq. (6)) common subtrees by extending from (e_1, e_2) and $\Delta'(v_1, v_2)$ (eq. (7)) common subtrees with root at (v_1, v_2).

Obviously $\Delta'(v_1, v_2)$ calculated by Algorithm 1 is a proper convolution kernel since it simply counts the number of common subtrees under the root (v_1, v_2). Therefore, $K_f(f_1, f_2)$ defined at eq. (5) and calculated through $\Delta'(v_1, v_2)$ is also a proper convolution kernel. From eq. (5) and Algorithm 1, we can see that each hyper-edge pair (e_1, e_2) is only visited at most one time in computing the forest kernel. Thus the time complexity for computing $K_f(f_1, f_2)$ is $O(|E_1| \cdot |E_2|)$, where E_1 and E_2 are the set of hyper-edges in forests f_1 and f_2, respectively. Given a forest and the best parse trees, the number of hyper-edges is only several times (normally <=3 after pruning) than that of tree nodes in the parse tree[3].

[2] Tree can be viewed as a special case of forest with only one hyper-edge attached to each tree node.

[3] Suppose there are *K* forest nodes in a forest, each node has *M* associated hyper-edges fan out and each hyper-edge has *N* children. Then the forest is capable of encoding $M^{\frac{K-1}{N-1}}$ parse trees at most (Zhang et al., 2009b).

Algorithm 1.

Input:

f_1, f_2: *two packed forests*

v_1, v_2: *any two nodes of* f_1 *and* f_2

Notation:

$I_{eq}(\cdot,\cdot)$: *defined at* eq. (5)

$nl(e_1)$: *number of leaf node of* e_1

$leaf(e_1, j)$: *the* j^{th} *leaf node of* e_1

Output: $\Delta'(v_1, v_2)$

1. $\Delta'(v_1, v_2) = 0$
2. **if** $v_1.label \neq v_2.label$ **exit**
3. **for** each hyper-edge e_1 attached to v_1 **do**
4. **for** each hyper-edge e_2 attached to v_2 **do**
5. **if** $I_{eq}(e_1, e_2) == 0$ **do**
6. **goto** line 3
7. **else do**
8. $\Delta''(e_1, e_2) = \prod_{j=1}^{nl(e_1)}\left(1 + \Delta'\left(leaf(e_1, j), leaf(e_2, j)\right)\right)$ (6)
9. $\Delta'(v_1, v_2) \mathrel{+}= \Delta''(e_1, e_2)$ (7)
10. **end if**
11. **end for**
12. **end for**

Same as tree kernel, forest kernel is running more efficiently in practice since only two nodes with the same label needs to be further processed (line 2 of Algorithm 1).

Now let us see how to integrate fractional counts into forest kernel. According to Algorithm 1 (eq. (7)), we have (e_1/e_2 are attached to v_1/v_2, respectively)

$$\Delta'(v_1, v_2) = \sum_{e_1=e_2} \Delta''(e_1, e_2)$$

Recall eq. (4), a fractional count consists of outside, inside and subtree probabilities. It is more straightforward to incorporate the outside and subtree probabilities since all the subtrees with roots at (v_1, v_2) share the same outside probability and each hyper-edge pair is only visited one time. Thus we can integrate the two probabilities into $\Delta'(v_1, v_2)$ as follows.

$$\Delta'(v_1, v_2) = \lambda \cdot \alpha(v_1) \cdot \alpha(v_2) \cdot \sum_{e_1=e_2}\left(P(e_1) \cdot P(e_2) \cdot \Delta''(e_1, e_2)\right) \quad (8)$$

where, following tree kernel, a decay factor $\lambda(0 < \lambda \leq 1)$ is also introduced in order to make the kernel value less variable with respect to the subtree sizes (Collins and Duffy, 2002). It functions like multiplying each feature value by λ^{size_i}, where $size_i$ is the number of hyper-edges in $subtree_i$.

The inside probability is only involved when a node does not need to be further expanded. The integer 1 at eq. (6) represents such case. So the inside probability is integrated into eq. (6) by replacing the integer 1 as follows.

$$\Delta''(e_1, e_2) = \prod_{j=1}^{nl(e_1)} \left(\beta\big(leaf(e_1, j)\big) \cdot \beta\big(leaf(e_2, j)\big) + \frac{\Delta'\big(leaf(e_1, j), leaf(e_2, j)\big)}{\alpha\big(leaf(e_1, j)\big) \cdot \alpha\big(leaf(e_2, j)\big)} \right) \quad (9)$$

where in the last expression the two outside probabilities $\alpha\big(leaf(e_1, j)\big)$ and $\alpha\big(leaf(e_2, j)\big)$ are removed. This is because $leaf(e_1, j)$ and $leaf(e_2, j)$ are not roots of the subtrees of being explored (only outside probabilities of the root of a subtree should be counted in its *fractional count*), and $\Delta'\big(leaf(e_1, j), leaf(e_2, j)\big)$ already contains the two outside probabilities of $leaf(e_1, j)$ and $leaf(e_2, j)$.

Referring to eq. (3), each fractional count needs to be normalized by $\alpha\beta(root(f))$. Since $\alpha\beta(root(f))$ is independent to each individual fractional count, we do the normalization outside the recursive function $\Delta''(e_1, e_2)$. Then we can re-formulize eq. (5) as

$$K_f(f_1, f_2) = < \phi(f_1), \phi(f_2) > = \frac{\left(\sum_{v_1 \in N_1} \sum_{v_2 \in N_2} \Delta'(v_1, v_2)\right)}{\alpha\beta\big(root(f_1)\big) \cdot \alpha\beta\big(root(f_2)\big)} \quad (10)$$

Finally, since the size of input forests is not constant, the forest kernel value is normalized using the following equation.

$$\widehat{K}_f(f_1, f_2) = \frac{K_f(f_1, f_2)}{\sqrt{K_f(f_1, f_1) \cdot K_f(f_2, f_2)}} \quad (11)$$

From the above discussion, we can see that the proposed forest kernel is defined together by eqs. (11), (10), (9) and (8). Thanks to the compact representation of trees in forest and the recursive nature of the kernel function, the introduction of fractional counts and normalization do not change the convolution property and the time complexity of the forest kernel. Therefore, the forest kernel $\widehat{K}_f(f_1, f_2)$ is still a proper convolution kernel with quadratic time complexity.

3.3 Comparison with previous work

To the best of our knowledge, this is the first work to address convolution kernel over packed parse forest.

Convolution tree kernel is a special case of the proposed forest kernel. From feature exploration viewpoint, although theoretically they explore the same subtree feature spaces (defined recursively by CFG parsing rules), their feature values are different. Forest encodes exponential number of trees. So the number of subtree instances extracted from a forest is exponential number of times greater than that from its corresponding parse tree. The significant difference of the amount of subtree instances makes the parameters learned from forests more reliable and also can help to address the data sparseness issue. To some degree, forest kernel can be viewed as a tree kernel with very powerful *back-off* mechanism. In addition, forest kernel is much more robust against parsing errors than tree kernel.

Aiolli *et al.* (2006; 2007) propose using Direct Acyclic Graphs (DAG) as a compact representation of tree kernel-based models. This can largely reduce the computational burden and storage requirements by sharing the common structures and feature vectors in the kernel-based model. There are a few other previous works done by generalizing convolution tree kernels (Kashima and Koyanagi, 2003; Moschitti, 2006; Zhang et al., 2007). However, all of these works limit themselves to single tree structure from modeling viewpoint in nature.

From a broad viewpoint, as suggested by one reviewer of the paper, we can consider the forest kernel as an alternative solution proposed for the general problem of noisy inference pipelines (eg. speech translation by composition of FSTs, machine translation by translating over 'lattices' of segmentations (Dyer *et al*., 2008) or using parse tree info for downstream applications in our cases) . Following this line, Bunescu (2008) and Finkel et al. (2006) are two typical related works done in reducing cascading noisy. However, our works are not overlapped with each other as there are two totally different solutions for the same general problem. In addition, the main motivation of this paper is also different from theirs.

4 Experiments

Forest kernel has a broad application potential in NLP. In this section, we verify the effectiveness of the forest kernel on two NLP applications, semantic role labeling (SRL) (Gildea, 2002) and relation extraction (RE) (ACE, 2002-2006).

In our experiments, SVM (Vapnik, 1998) is selected as our classifier and the *one vs. others* strategy is adopted to select the one with the

largest margin as the final answer. In our implementation, we use the binary SVMLight (Joachims, 1998) and borrow the framework of the Tree Kernel Tools (Moschitti, 2004) to integrate our forest kernel into the SVMLight. We modify Charniak parser (Charniak, 2001) to output a packed forest. Following previous forest-based studies (Charniak and Johnson, 2005), we use the marginal probabilities of hyper-edges (i.e., the Viterbi-style inside-outside probabilities and set the pruning threshold as 8) for forest pruning.

4.1 Semantic role labeling

Given a sentence and each predicate (either a target verb or a noun), SRL recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles, e.g., A0 for *Agent*, A1 for *Patient* …) of the predicate or *non*-argument. We use the CoNLL-2005 shared task on Semantic Role Labeling (Carreras and Màrquez, 2005) for the evaluation of our forest kernel method. To speed up the evaluation process, the same as Che *et al.* (2008), we use a subset of the entire training corpus (WSJ sections 02-05 of the entire sections 02-21) for training, section 24 for development and section 23 for test, where there are 35 roles including 7 Core (A0–A5, AA), 14 Adjunct (AM-) and 14 Reference (R-) arguments.

The state-of-the-art SRL methods (Carreras and Màrquez, 2005) use constituents as the labeling units to form the labeled arguments. Due to the errors from automatic parsing, it is impossible for all arguments to find their matching constituents in the single 1-best parse trees. Statistics on the training data shows that 9.78% of arguments have no matching constituents using the Charniak parser (Charniak, 2001), and the number increases to 11.76% when using the Collins parser (Collins, 1999). In our method, we break the limitation of 1-best parse tree and regard each span rooted by a single forest node (i.e., a *sub-forest* with one or more roots) as a candidate argument. This largely reduces the unmatched arguments from 9.78% to 1.31% after forest pruning. However, it also results in a very large amount of argument candidates that is 5.6 times as many as that from 1-best tree. Fortunately, after the pre-processing stage of argument pruning (Xue and Palmer, 2004)[4], although the amount of unmatched argument increases a little bit to 3.1%, its generated total candidate amount decreases substantially to only 1.31 times of that from 1-best parse tree. This clearly shows the advantages of the forest-based method over tree-based in SRL.

The best-reported tree kernel method for SRL $K_{hybrid} = \theta \cdot K_{path} + (1-\theta) \cdot K_{cs}$ $(0 \le \theta \le 1)$, proposed by Che et al. (2006)[5], is adopted as our baseline kernel. We implemented the K_{hybrid} in tree case ($K_{T-hybrid}$, using tree kernel to compute K_{path} and K_{cs}) and in forest case ($K_{F-hybrid}$, using tree kernel to compute K_{path} and K_{cs}).

	Precision	**Recall**	**F-Score**
$K_{T-hybrid}$ (Tree)	76.02	67.38	71.44
$K_{F-hybrid}$ (Forest)	79.06	69.12	73.76

Table 1: Performance comparison of SRL (%)

Table 1 shows that the forest kernel significantly outperforms (χ^2 test with p=0.01) the tree kernel with an absolute improvement of 2.32 (73.76-71.42) percentage in F-Score, representing a relative error rate reduction of 8.19% (2.32/(100-71.64)). This convincingly demonstrates the advantage of the forest kernel over the tree kernel. It suggests that the structured features represented by subtree are very useful to SRL. The performance improvement is mainly due to the fact that forest encodes much more such structured features and the forest kernel is able to more effectively capture such structured features than the tree kernel. Besides F-Score, both precision and recall also show significantly improvement (χ^2 test with p=0.01). The reason for recall improvement is mainly due to the lower rate of unmatched argument (3.1% only) with only a little bit overhead (1.31 times) (see the previous discussion in this section). The precision improvement is mainly attributed to fact that we use *sub-forest* to represent argument instances, rather than sub-tree used in tree kernel, where the sub-tree is only one tree encoded in the *sub-forest*.

[4] We extend (Xue and Palmer, 2004)'s argument pruning algorithm from tree-based to forest-based. The algorithm is very effective. It can prune out around 90% argument candidates in parse tree-based SRL and thus makes the amounts of positive and negative training instances (arguments) more balanced. We apply the same pruning strategies to forest plus our heuristic rules to prune out some of the arguments with span overlapped with each other and those arguments with very small inside probabilities, depending on the numbers of candidates in the span.

[5] K_{path} and K_{cs} are two standard convolution tree kernels to describe predicate-argument path substructures and argument syntactic substructures, respectively.

4.2 Relation extraction

As a subtask of information extraction, relation extraction is to extract various semantic relations between entity pairs from text. For example, the sentence "Bill Gates is chairman and chief software architect of Microsoft Corporation" conveys the semantic relation "EMPLOYMENT.executive" between the entities "Bill Gates" (person) and "Microsoft Corporation" (company). We adopt the method reported in Zhang et al. (2006) as our baseline method as it reports the state-of-the-art performance using tree kernel-based composite kernel method for RE. We replace their tree kernels with our forest kernels and use the same experimental settings as theirs. We carry out the same five-fold cross validation experiment on the same subset of ACE 2004 data (LDC2005T09, ACE 2002-2004) as that in Zhang et al. (2006). The data contain 348 documents and 4400 relation instances.

In SRL, constituents are used as the labeling units to form the labeled arguments. However, previous work (Zhang et al., 2006) shows that if we use complete constituent (MCT) as done in SRL to represent relation instance, there is a large performance drop compared with using the path-enclosed tree (PT)[6]. By simulating PT, we use the minimal fragment of a forest covering the two entities and their internal words to represent a relation instance by only parsing the span covering the two entities and their internal words.

	Precision	Recall	F-Score
Zhang et al. (2006):Tree	68.6	59.3	63.6
Ours: Forest	70.3	60.0	64.7

Table 2: Performance Comparison of RE (%) over 23 subtypes on the ACE 2004 data

Table 2 compares the performance of the forest kernel and the tree kernel on relation extraction. We can see that the forest kernel significantly outperforms (χ^2 test with p=0.05) the tree kernel by 1.1 point of F-score. This further verifies the effectiveness of the forest kernel method for modeling NLP structured data. In summary, we further observe the high precision improvement that is consistent with the SRL experiments. However, the recall improvement is not as significant as observed in SRL. This is because unlike SRL, RE has no un-matching issues in generating relation instances. Moreover, we find that the performance improvement in RE is not as good as that in SRL. Although we know that performance is task-dependent, one of the possible reasons is that SRL tends to be long-distance grammatical structure-related while RE is local and semantic-related as observed from the two experimental benchmark data.

[6] MCT is the minimal constituent rooted by the nearest common ancestor of the two entities under consideration while PT is the minimal portion of the parse tree (may not be a complete subtree) containing the two entities and their internal lexical words. Since in many cases, the two entities and their internal words cannot form a grammatical constituent, MCT may introduce too many noisy context features and thus lead to the performance drop.

5 Conclusions and Future Work

Many NLP applications have benefited from the success of convolution kernel over parse tree. Since a packed parse forest contains much richer structured features than a parse tree, we are motivated to develop a technology to measure the syntactic similarity between two forests.

To achieve this goal, in this paper, we design a convolution kernel over packed forest by generalizing the tree kernel. We analyze the object space of the forest kernel, the fractional count for feature value computing and design a dynamic programming algorithm to realize the forest kernel with quadratic time complexity. Compared with the tree kernel, the forest kernel is more robust against parsing errors and data sparseness issues. Among the broad potential NLP applications, the problems in SRL and RE provide two pointed scenarios to verify our forest kernel. Experimental results demonstrate the effectiveness of the proposed kernel in structured NLP data modeling and the advantages over tree kernel.

In the future, we would like to verify the forest kernel in more NLP applications. In addition, as suggested by one reviewer, we may consider rescaling the probabilities (exponentiating them by a constant value) that are used to compute the fractional counts. We can sharpen or flatten the distributions. This basically says "how seriously do we want to take the very best derivation" compared to the rest. However, the challenge is that we compute the fractional counts together with the forest kernel recursively by using the Inside-Outside probabilities. We cannot differentiate the individual parse tree's contribution to a fractional count on the fly. One possible solution is to do the probability rescaling off-line before kernel calculation. This would be a very interesting research topic of our future work.

References

ACE (2002-2006). *The Automatic Content Extraction Projects*. http://www.ldc.upenn.edu/Projects/ACE/

Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti and Alessandro Moschitti. 2006. *Fast Online Kernel Learning for Trees*. ICDM-2006

Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti and Alessandro Moschitti. 2007. *Efficient Kernel-based Learning for Trees*. IEEE Symposium on Computational Intelligence and Data Mining (CIDM-2007)

J. Baker. 1979. *Trainable grammars for speech recognition*. The 97th meeting of the Acoustical Society of America

S. Billot and S. Lang. 1989. *The structure of shared forest in ambiguous parsing*. ACL-1989

Razvan Bunescu. 2008. *Learning with Probabilistic Features for Improved Pipeline Models*. EMNLP-2008

X. Carreras and Lluıs Màrquez. 2005. *Introduction to the CoNLL-2005 shared task: SRL*. CoNLL-2005

E. Charniak. 2001. *Immediate-head Parsing for Language Models*. ACL-2001

E. Charniak and Mark Johnson. 2005. *Corse-to-fine-grained n-best parsing and discriminative reranking*. ACL-2005

Wanxiang Che, Min Zhang, Ting Liu and Sheng Li. 2006. *A hybrid convolution tree kernel for semantic role labeling*. COLING-ACL-2006 (poster)

WanXiang Che, Min Zhang, Aiti Aw, Chew Lim Tan, Ting Liu and Sheng Li. 2008. *Using a Hybrid Convolution Tree Kernel for Semantic Role Labeling*. ACM Transaction on Asian Language Information Processing

M. Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. dissertation, Pennsylvania University

M. Collins and N. Duffy. 2002. *Convolution Kernels for Natural Language*. NIPS-2002

Christopher Dyer, Smaranda Muresan and Philip Resnik. 2008. *Generalizing Word Lattice Translation.* ACL-HLT-2008

Jenny Rose Finkel, Christopher D. Manning and Andrew Y. Ng. 2006. *Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines.* EMNLP-2006

Y. Freund and R. E. Schapire. 1999. *Large margin classification using the perceptron algorithm*. Machine Learning, 37(3):277-296

D. Guldea. 2002. *Probabilistic models of verb-argument structure*. COLING-2002

D. Haussler. 1999. *Convolution Kernels on Discrete Structures.* Technical Report UCS-CRL-99-10, University of California, Santa Cruz

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

Karim Lari and Steve J. Young. 1990. *The estimation of stochastic context-free grammars using the inside-outside algorithm*. Computer Speech and Language. 4(35–56)

H. Kashima and T. Koyanagi. 2003. *Kernels for Semi-Structured Data*. ICML-2003

Dan Klein and Christopher D. Manning. 2001. *Parsing and Hypergraphs*. IWPT-2001

T. Joachims. 1998. *Text Categorization with Support Vecor Machine: learning with many relevant features*. ECML-1998

Haitao Mi and Liang Huang. 2008. *Forest-based Translation Rule Extraction*. EMNLP-2008

Alessandro Moschitti. 2004. *A Study on Convolution Kernels for Shallow Semantic Parsing*. ACL-2004

Alessandro Moschitti. 2006. *Syntactic kernels for natural language learning: the semantic role labeling case*. HLT-NAACL-2006 (short paper)

Martha Palmer, Dan Gildea and Paul Kingsbury. 2005. *The proposition bank: An annotated corpus of semantic roles*. Computational Linguistics. 31(1)

F. Rosenblatt. 1962. *Principles of Neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, Washington D.C.

Masaru Tomita. 1987. *An Efficient Augmented-Context-Free Parsing Algorithm*. Computational Linguistics 13(1-2): 31-46

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley

C. Watkins. 1999. *Dynamic alignment kernels*. In A. J. Smola, B. Sch ̈olkopf, P. Bartlett, and D. Schuurmans (Eds.), Advances in kernel methods. MIT Press

Nianwen Xue and Martha Palmer. 2004. *Calibrating features for semantic role labeling.* EMNLP-2004

Xiaofeng Yang, Jian Su and Chew Lim Tan. 2006. *Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge*. COLING-ACL-2006

Dell Zhang and W. Lee. 2003. *Question classification using support vector machines*. SIGIR-2003

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw and Chew Lim Tan. 2009a. *Forest-based Tree Sequence to String Translation Model*. ACL-IJCNLP-2009

Hui Zhang, Min Zhang, Haizhou Li and Chew Lim Tan. 2009b. *Fast Translation Rule Matching for*

Syntax-based Statistical Machine Translation. EMNLP-2009

Min Zhang, Jie Zhang, Jian Su and GuoDong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*. COLING-ACL-2006

Min Zhang, W. Che, A. Aw, C. Tan, G. Zhou, T. Liu and S. Li. 2007. *A Grammar-driven Convolution Tree Kernel for Semantic Role Classification*. ACL-2007

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan and Sheng Li. 2008. *A Tree Sequence Alignment-based Tree-to-Tree Translation Model*. ACL-2008

Min Zhang and Haizhou Li. 2009. *Tree Kernel-based SVM with Structured Syntactic Knowledge for BTG-based Phrase Reordering*. EMNLP-2009

Estimating Strictly Piecewise Distributions

Jeffrey Heinz
University of Delaware
Newark, Delaware, USA
heinz@udel.edu

James Rogers
Earlham College
Richmond, Indiana, USA
jrogers@quark.cs.earlham.edu

Abstract

Strictly Piecewise (SP) languages are a subclass of regular languages which encode certain kinds of long-distance dependencies that are found in natural languages. Like the classes in the Chomsky and Subregular hierarchies, there are many independently converging characterizations of the SP class (Rogers et al., to appear). Here we define SP distributions and show that they can be efficiently estimated from positive data.

1 Introduction

Long-distance dependencies in natural language are of considerable interest. Although much attention has focused on long-distance dependencies which are beyond the expressive power of models with finitely many states (Chomsky, 1956; Joshi, 1985; Shieber, 1985; Kobele, 2006), there are some long-distance dependencies in natural language which permit finite-state characterizations. For example, although it is well-known that vowel and consonantal harmony applies across any arbitrary number of intervening segments (Ringen, 1988; Baković, 2000; Hansson, 2001; Rose and Walker, 2004) and that phonological patterns are regular (Johnson, 1972; Kaplan and Kay, 1994), it is less well-known that harmony patterns are largely characterizable by the Strictly Piecewise languages, a subregular class of languages with independently-motivated, converging characterizations (see Heinz (2007, to appear) and especially Rogers et al. (2009)).

As shown by Rogers et al. (to appear), the Strictly Piecewise (SP) languages, which make distinctions on the basis of (potentially) discontiguous subsequences, are precisely analogous to the Strictly Local (SL) languages (McNaughton and Papert, 1971; Rogers and Pullum, to appear), which make distinctions on the basis of contiguous subsequences. The Strictly Local languages are the formal-language theoretic foundation for n-gram models (Garcia et al., 1990), which are widely used in natural language processing (NLP) in part because such distributions can be estimated from positive data (i.e. a corpus) (Jurafsky and Martin, 2008). N-gram models describe probability distributions over all strings on the basis of the Markov assumption (Markov, 1913): that the probability of the next symbol only depends on the previous contiguous sequence of length $n-1$. From the perspective of formal language theory, these distributions are perhaps properly called Strictly k-Local distributions (SL_k) where $k = n$. It is well-known that one limitation of the Markov assumption is its inability to express any kind of long-distance dependency.

This paper defines Strictly k-Piecewise (SP_k) distributions and shows how they too can be efficiently estimated from positive data. In contrast with the Markov assumption, our assumption is that the probability of the next symbol is conditioned on the previous set of discontiguous subsequences of length $k-1$ in the string. While this suggests the model has too many parameters (one for each subset of all possible subsequences), in fact the model has on the order of $|\Sigma|^{k+1}$ parameters because of an independence assumption: there is no interaction between different subsequences. As a result, SP distributions are efficiently computable even though they condition the probability of the next symbol on the occurrences of earlier (possibly very distant) discontiguous subsequences. Essentially, these SP distributions reflect a kind of long-term memory.

On the other hand, SP models have no short-term memory and are unable to make distinctions on the basis of contiguous subsequences. We do not intend SP models to replace n-gram models, but instead expect them to be used alongside of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 886–896,
Uppsala, Sweden, 11-16 July 2010.

them. Exactly how this is to be done is beyond the scope of this paper and is left for future research.

Since SP languages are the analogue of SL languages, which are the formal-language theoretical foundation for n-gram models, which are widely used in NLP, it is expected that SP distributions and their estimation will also find wide application. Apart from their interest to problems in theoretical phonology such as phonotactic learning (Coleman and Pierrehumbert, 1997; Hayes and Wilson, 2008; Heinz, to appear), it is expected that their use will have application, in conjunction with n-gram models, in areas that currently use them; e.g. augmentative communication (Newell et al., 1998), part of speech tagging (Brill, 1995), and speech recognition (Jelenik, 1997).

§2 provides basic mathematical notation. §3 provides relevant background on the subregular hierarchy. §4 describes automata-theoretic characterizations of SP languages. §5 defines SP distributions. §6 shows how these distributions can be efficiently estimated from positive data and provides a demonstration. §7 concludes the paper.

2 Preliminaries

We start with some mostly standard notation. Σ denotes a finite set of symbols and a string over Σ is a finite sequence of symbols drawn from that set. Σ^k, $\Sigma^{\leq k}$, $\Sigma^{\geq k}$, and Σ^* denote all strings over this alphabet of length k, of length less than or equal to k, of length greater than or equal to k, and of any finite length, respectively. ϵ denotes the empty string. $|w|$ denotes the length of string w. The prefixes of a string w are $\text{Pfx}(w) = \{v : \exists u \in \Sigma^* \text{ such that } vu = w\}$. When discussing partial functions, the notation $\uparrow$ and $\downarrow$ indicates that the function is undefined, respectively is defined, for particular arguments.

A *language* L is a subset of Σ^*. A *stochastic language* $\mathcal{D}$ is a probability distribution over Σ^*. The probability p of word w with respect to $\mathcal{D}$ is written $Pr_{\mathcal{D}}(w) = p$. Recall that all distributions $\mathcal{D}$ must satisfy $\sum_{w \in \Sigma^*} Pr_{\mathcal{D}}(w) = 1$. If L is language then $Pr_{\mathcal{D}}(L) = \sum_{w \in L} Pr_{\mathcal{D}}(w)$.

A *Deterministic Finite-state Automaton* (DFA) is a tuple $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F \rangle$ where Q is the state set, Σ is the alphabet, q_0 is the start state, δ is a deterministic transition function with domain $Q \times \Sigma$ and codomain Q, F is the set of accepting states. Let $\hat{d} : Q \times \Sigma^* \rightarrow Q$ be the (partial) path function of $\mathcal{M}$, i.e., $\hat{d}(q, w)$ is the (unique) state reachable from state q via the sequence w, if any, or $\hat{d}(q, w)\uparrow$ otherwise. The language recognized by a DFA $\mathcal{M}$ is $L(\mathcal{M}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{d}(q_0, w)\downarrow \in F\}$.

A state is *useful* iff for all $q \in Q$, there exists $w \in \Sigma^*$ such that $\delta(q_0, w) = q$ and there exists $w \in \Sigma^*$ such that $\delta(q, w) \in F$. *Useless* states are not useful. DFAs without useless states are *trimmed*.

Two strings w and v over Σ are *distinguished* by a DFA $\mathcal{M}$ iff $\hat{d}(q_0, w) \neq \hat{d}(q_0, v)$. They are *Nerode equivalent* with respect to a language L if and only if $wu \in L \iff vu \in L$ for all $u \in \Sigma^*$. All DFAs which recognize L must distinguish strings which are inequivalent in this sense, but no DFA recognizing L necessarily distinguishes any strings which are equivalent. Hence the number of equivalence classes of strings over Σ modulo Nerode equivalence with respect to L gives a (tight) lower bound on the number of states required to recognize L.

A DFA is *minimal* if the size of its state set is minimal among DFAs accepting the same language. The *product* of n DFAs $\mathcal{M}_1 \ldots \mathcal{M}_n$ is given by the standard construction over the state space $Q_1 \times \ldots \times Q_n$ (Hopcroft et al., 2001).

A *Probabilistic Deterministic Finite-state Automaton* (PDFA) is a tuple $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F, T \rangle$ where Q is the state set, Σ is the alphabet, q_0 is the start state, δ is a deterministic transition function, F and T are the final-state and transition probabilities. In particular, $T : Q \times \Sigma \rightarrow \mathbb{R}^+$ and $F : Q \rightarrow \mathbb{R}^+$ such that

$$\text{for all } q \in Q,\ F(q) + \sum_{a \in \Sigma} T(q, a) = 1. \quad (1)$$

Like DFAs, for all $w \in \Sigma^*$, there is at most one state reachable from q_0. PDFAs are typically represented as labeled directed graphs as in Figure 1.

A PDFA $\mathcal{M}$ generates a stochastic language $\mathcal{D}_{\mathcal{M}}$. If it exists, the (unique) *path* for a word $w = a_0 \ldots a_k$ belonging to Σ^* through a PDFA is a sequence $\langle (q_0, a_0), (q_1, a_1), \ldots, (q_k, a_k) \rangle$, where $q_{i+1} = \delta(q_i, a_i)$. The probability a PDFA assigns to w is obtained by multiplying the transition probabilities with the final probability along w's path if

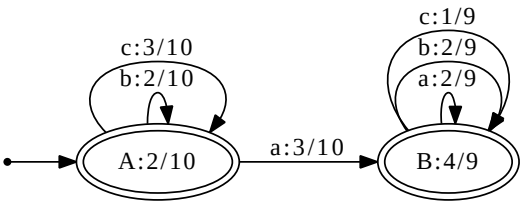


Figure 1: A picture of a PDFA with states labeled A and B. The probabilities of T and F are located to the right of the colon.

it exists, and zero otherwise.

$$Pr_{\mathcal{D}_\mathcal{M}}(w) = \left(\prod_{i=1}^{k} T(q_{i-1}, a_{i-1})\right) \cdot F(q_{k+1}) \quad (2)$$

if $\hat{d}(q_0, w)\downarrow$ and 0 otherwise

A probability distribution is *regular deterministic* iff there is a PDFA which generates it.

The *structural components* of a PDFA $\mathcal{M}$ are its states Q, its alphabet Σ, its transitions δ, and its initial state q_0. By *structure* of a PDFA, we mean its structural components. Each PDFA $\mathcal{M}$ defines a family of distributions given by the possible instantiations of T and F satisfying Equation 1. These distributions have $|Q| \cdot (|\Sigma| + 1)$ independent parameters (since for each state there are $|\Sigma|$ possible transitions plus the possibility of finality.)

We define the product of PDFA in terms of *co-emission probabilities* (Vidal et al., 2005a).

Definition 1 *Let $\mathcal{A}$ be a vector of PDFAs and let $|\mathcal{A}| = n$. For each $1 \leq i \leq n$ let $\mathcal{M}_i = \langle Q_i, \Sigma, q_{0i}, \delta_i, F_i, T_i \rangle$ be the ith PDFA in $\mathcal{A}$. The probability that σ is co-emitted from $q_1, \ldots, q_n$ in $Q_1, \ldots, Q_n$, respectively, is*

$$CT(\langle \sigma, q_1 \ \ldots \ q_n \rangle) = \prod_{i=1}^{n} T_i(q_i, \sigma).$$

Similarly, the probability that a word simultaneously ends at $q_1 \in Q_1 \ \ldots \ q_n \in Q_n$ is

$$CF(\langle q_1 \ \ldots \ q_n \rangle) = \prod_{i=1}^{n} F_i(q_i).$$

Then $\bigotimes \mathcal{A} = \langle Q, \Sigma, q_0, \delta, F, T \rangle$ where

1. *Q, q_0, and δ are defined as with DFA product.*
2. *For all $\langle q_1 \ \ldots \ q_n \rangle \in Q$, let $Z(\langle q_1 \ \ldots \ q_n \rangle) =$*

$$CF(\langle q_1 \ \ldots \ q_n \rangle) + \sum_{\sigma \in \Sigma} CT(\langle \sigma, q_1 \ \ldots \ q_n \rangle)$$

be the normalization term*; and*

   (a) *let $F(\langle q_1 \ \ldots \ q_n \rangle) = \frac{CF(\langle q_1 \ \ldots \ q_n \rangle)}{Z(\langle q_1 \ \ldots \ q_n \rangle)}$; and*

   (b) *for all $\sigma \in \Sigma$, let $T(\langle q_1 \ \ldots \ q_n \rangle, \sigma) = \frac{CT(\langle \sigma, q_1 \ \ldots \ q_n \rangle)}{Z(\langle q_1 \ \ldots \ q_n \rangle)}$*

In other words, the numerators of T and F are defined to be the co-emission probabilities (Vidal et al., 2005a), and division by Z ensures that $\mathcal{M}$ defines a well-formed probability distribution. Statistically speaking, the co-emission product makes an independence assumption: the probability of σ being co-emitted from $q_1, \ldots, q_n$ is exactly what one expects if there is no interaction between the individual factors; that is, between the probabilities of σ being emitted from any q_i. Also note order of product is irrelevant up to renaming of the states, and so therefore we also speak of taking the product of a set of PDFAs (as opposed to an ordered vector).

Estimating regular deterministic distributions is well-studied problem (Vidal et al., 2005a; Vidal et al., 2005b; de la Higuera, in press). We limit discussion to cases when the structure of the PDFA is known. Let S be a finite sample of words drawn from a regular deterministic distribution $\mathcal{D}$. The problem is to estimate parameters T and F of $\mathcal{M}$ so that $\mathcal{D}_\mathcal{M}$ approaches $\mathcal{D}$. We employ the widely-adopted maximum likelihood (ML) criterion for this estimation.

$$(\hat{T}, \hat{F}) = \underset{T,F}{\operatorname{argmax}} \left(\prod_{w \in S} Pr_\mathcal{M}(w)\right) \quad (3)$$

It is well-known that if $\mathcal{D}$ is generated by some PDFA $\mathcal{M}'$ with the same structural components as $\mathcal{M}$, then optimizing the ML estimate guarantees that $\mathcal{D}_\mathcal{M}$ approaches $\mathcal{D}$ as the size of S goes to infinity (Vidal et al., 2005a; Vidal et al., 2005b; de la Higuera, in press).

The optimization problem (3) is simple for deterministic automata with known structural components. Informally, the corpus is passed through the PDFA, and the paths of each word through the corpus are tracked to obtain counts, which are then normalized by state. Let $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F, T \rangle$ be the PDFA whose parameters F and T are to be estimated. For all states $q \in Q$ and symbols $a \in \Sigma$, The ML estimation of the probability of $T(q, a)$ is obtained by dividing the number of times this transition is used in parsing the sample S by the

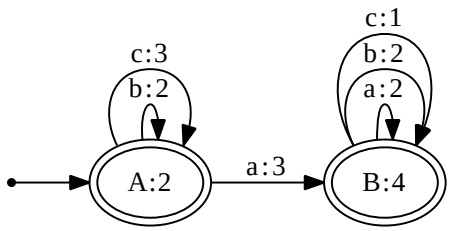


Figure 2: The automata shows the counts obtained by parsing $\mathcal{M}$ with sample $S = \{ab, bba, \epsilon, cab, acb, cc\}$.

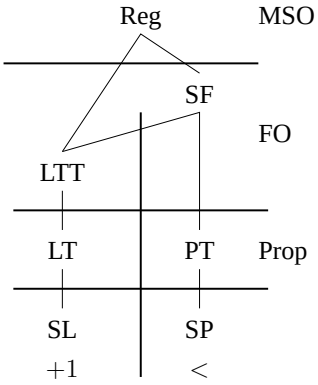


Figure 3: Parallel Sub-regular Hierarchies.

number of times state q is encountered in the parsing of S. Similarly, the ML estimation of $F(q)$ is obtained by calculating the relative frequency of state q being final with state q being encountered in the parsing of S. For both cases, the division is *normalizing*; i.e. it guarantees that there is a well-formed probability distribution at each state. Figure 2 illustrates the counts obtained for a machine $\mathcal{M}$ with sample $S = \{ab, bba, \epsilon, cab, acb, cc\}$.[1] Figure 1 shows the PDFA obtained after normalizing these counts.

3 Subregular Hierarchies

Within the class of regular languages there are dual hierarchies of language classes (Figure 3), one in which languages are defined in terms of their *contiguous substrings* (up to some length k, known as k*-factors*), starting with the languages that are *Locally Testable in the Strict Sense* (SL), and one in which languages are defined in terms of their not necessarily contiguous *subsequences*, starting with the languages that are *Piecewise Testable in the Strict Sense* (SP). Each language class in these hierarchies has independently motivated, converging characterizations and each has been claimed to correspond to specific, fundamental cognitive capabilities (McNaughton and Papert, 1971; Brzozowski and Simon, 1973; Simon, 1975; Thomas, 1982; Perrin and Pin, 1986; García and Ruiz, 1990; Beauquier and Pin, 1991; Straubing, 1994; García and Ruiz, 1996; Rogers and Pullum, to appear; Kontorovich et al., 2008; Rogers et al., to appear).

Languages in the weakest of these classes are defined only in terms of the set of factors (SL) or subsequences (SP) which are licensed to occur in the string (equivalently the complement of that set with respect to $\Sigma^{\leq k}$, the *forbidden factors* or *forbidden subsequences*). For example, the set containing the forbidden 2-factors $\{ab, ba\}$ defines a Strictly 2-Local language which includes all strings except those with contiguous substrings $\{ab, ba\}$. Similarly since the parameters of n-gram models (Jurafsky and Martin, 2008) assign probabilities to symbols given the preceding contiguous substrings up to length $n-1$, we say they describe Strictly n-Local distributions.

These hierarchies have a very attractive model-theoretic characterization. The *Locally Testable* (LT) and *Piecewise Testable* languages are exactly those that are definable by propositional formulae in which the atomic formulae are blocks of symbols interpreted factors (LT) or subsequences (PT) of the string. The languages that are testable in the strict sense (SL and SP) are exactly those that are definable by formulae of this sort restricted to conjunctions of negative literals. Going the other way, the languages that are definable by First-Order formulae with adjacency (successor) but not precedence (less-than) are exactly the *Locally Threshold Testable* (LTT) languages. The *Star-Free* languages are those that are First-Order definable with precedence alone (adjacency being FO definable from precedence). Finally, by extending to Monadic Second-Order formulae (with either signature, since they are MSO definable from each other), one obtains the full class of Regular languages (McNaughton and Papert, 1971; Thomas, 1982; Rogers and Pullum, to appear; Rogers et al., to appear).

The relation between strings which is fundamental along the Piecewise branch is the *subse-*

[1] Technically, this acceptor is neither a simple DFA or PDFA; rather, it has been called a Frequency DFA. We do not formally define them here, see (de la Higuera, in press).

quence relation, which is a partial order on Σ^*:

$$w \sqsubseteq v \overset{\text{def}}{\iff} w = \varepsilon \text{ or } w = \sigma_1 \cdots \sigma_n \text{ and}$$
$$(\exists w_0, \ldots, w_n \in \Sigma^*)[v = w_0\sigma_1 w_1 \cdots \sigma_n w_n].$$

in which case we say w is a *subsequence* of v.

For $w \in \Sigma^*$, let

$$\mathrm{P}_k(w) \overset{\text{def}}{=} \{v \in \Sigma^k \mid v \sqsubseteq w\} \text{ and}$$
$$\mathrm{P}_{\leq k}(w) \overset{\text{def}}{=} \{v \in \Sigma^{\leq k} \mid v \sqsubseteq w\},$$

the set of subsequences of length k, respectively length no greater than k, of w. Let $\mathrm{P}_k(L)$ and $\mathrm{P}_{\leq k}(L)$ be the natural extensions of these to sets of strings. Note that $\mathrm{P}_0(w) = \{\varepsilon\}$, for all $w \in \Sigma^*$, that $\mathrm{P}_1(w)$ is the set of symbols occurring in w and that $\mathrm{P}_{\leq k}(L)$ is finite, for all $L \subseteq \Sigma^*$.

Similar to the Strictly Local languages, Strictly Piecewise languages are defined only in terms of the set of subsequences (up to some length k) which are licensed to occur in the string.

Definition 2 (SP_k **Grammar,** SP**)** *A* SP_k *grammar is a pair* $\mathcal{G} = \langle \Sigma, G \rangle$ *where* $G \subseteq \Sigma^k$. *The language licensed by a* SP_k *grammar is*

$$L(\mathcal{G}) \overset{\text{def}}{=} \{w \in \Sigma^* \mid P_{\leq k}(w) \subseteq P_{\leq k}(G)\}.$$

A language is SP_k iff it is $L(\mathcal{G})$ for some SP_k grammar $\mathcal{G}$. It is SP iff it is SP_k for some k.

This paper is primarily concerned with estimating Strictly Piecewise distributions, but first we examine in greater detail properties of SP languages, in particular DFA representations.

4 DFA representations of SP Languages

Following Sakarovitch and Simon (1983), Lothaire (1997) and Kontorovich, et al. (2008), we call the set of strings that contain w as a subsequence the *principal shuffle ideal*[2] of w:

$$\mathrm{SI}(w) = \{v \in \Sigma^* \mid w \sqsubseteq v\}.$$

The *shuffle ideal* of a set of strings is defined as

$$\mathrm{SI}(S) = \cup_{w \in S} \mathrm{SI}(w)$$

Rogers et al. (to appear) establish that the SP languages have a variety of characteristic properties.

Theorem 1 *The following are equivalent:*[3]

[2]Properly $\mathrm{SI}(w)$ is the principal ideal generated by $\{w\}$ wrt the inverse of $\sqsubseteq$.

[3]For a complete proof, see Rogers et al. (to appear). We only note that 5 implies 1 by DeMorgan's theorem and the fact that every shuffle ideal is finitely generated (see also Lothaire (1997)).

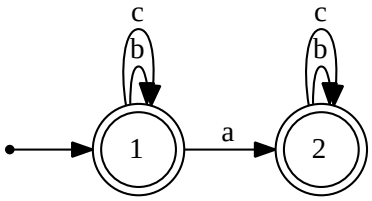


Figure 4: The DFA representation of $\overline{\mathrm{SI}(aa)}$.

1. $L = \bigcap_{w \in S} [\overline{\mathrm{SI}(w)}],\ S$ *finite,*
2. $L \in$ SP
3. $(\exists k)[P_{\leq k}(w) \subseteq P_{\leq k}(L) \Rightarrow w \in L]$,
4. $w \in L$ *and* $v \sqsubseteq w \Rightarrow v \in L$ *(*L *is* subsequence closed*),*
5. $L = \overline{\mathrm{SI}(X)},\ X \subseteq \Sigma^*$ *(*L *is the complement of a shuffle ideal).*

The DFA representation of the complement of a shuffle ideal is especially important.

Lemma 1 *Let* $w \in \Sigma^k$, $w = \sigma_1 \cdots \sigma_k$, *and* $\mathcal{M}_{\overline{\mathrm{SI}(w)}} = \langle Q, \Sigma, q_0, \delta, F \rangle$, *where* $Q = \{i \mid 1 \leq i \leq k\}$, $q_0 = 1$, $F = Q$ *and for all* $q_i \in Q, \sigma \in \Sigma$*:*

$$\delta(q_i, \sigma) = \begin{cases} q_{i+1} & \text{if } \sigma = \sigma_i \text{ and } i < k, \\ \uparrow & \text{if } \sigma = \sigma_i \text{ and } i = k, \\ q_i & \text{otherwise.} \end{cases}$$

Then $\mathcal{M}_{\overline{\mathrm{SI}(w)}}$ *is a minimal, trimmed DFA that recognizes the complement of* $\mathrm{SI}(w)$*, i.e.,* $\overline{\mathrm{SI}(w)} = L(\mathcal{M}_{\overline{\mathrm{SI}(w)}})$.

Figure 4 illustrates the DFA representation of the complement of $\mathrm{SI}(aa)$ with $\Sigma = \{a, b, c\}$. It is easy to verify that the machine in Figure 4 accepts all and only those words which do not contain an aa subsequence.

For any SP_k language $L = L(\langle \Sigma, G \rangle) \neq \Sigma^*$, the first characterization (1) in Theorem 1 above yields a non-deterministic finite-state representation of L, which is a set $\mathcal{A}$ of DFA representations of complements of principal shuffle ideals of the elements of G. The trimmed automata product of this set yields a DFA, with the properties below (Rogers et al., to appear).

Lemma 2 *Let* $\mathcal{M}$ *be a trimmed DFA recognizing a* SP_k *language constructed as described above. Then:*

1. *All states of* $\mathcal{M}$ *are accepting states:* $F = Q$.

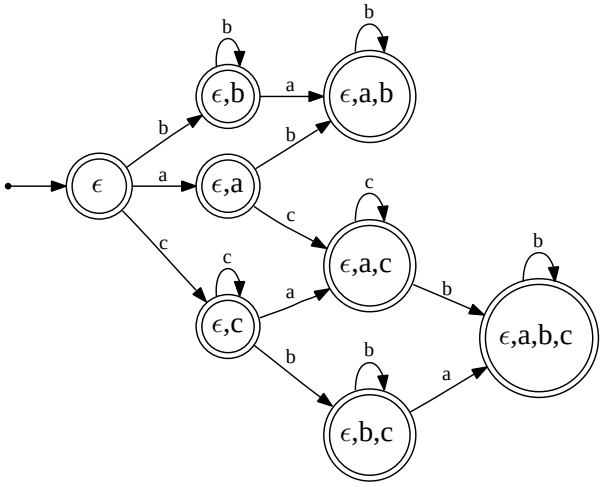


Figure 5: The DFA representation of the of the SP language given by $\mathcal{G} = \langle\{a, b, c\}, \overline{\{aa, bc\}}\rangle$. Names of the states reflect subsets of subsequences up to length 1 of prefixes of the language. Note this DFA is trimmed, but not minimal.

2. *For all $q_1, q_2 \in Q$ and $\sigma \in \Sigma$, if $\hat{d}(q_1, \sigma)\uparrow$ and $\hat{d}(q_1, w) = q_2$ for some $w \in \Sigma^*$ then $\hat{d}(q_2, \sigma)\uparrow$. (Missing edges propagate down.)*

Figure 5 illustrates with the DFA representation of the of the SP_2 language given by $\mathcal{G} = \langle\{a, b, c\}, \overline{\{aa, bc\}}\rangle$. It is straightforward to verify that this DFA is identical (modulo relabeling of state names) to one obtained by the trimmed product of the DFA representations of the complement of the principal shuffle ideals of aa and bc, which are the prohibited subsequences.

States in the DFA in Figure 5 correspond to the subsequences up to length 1 of the prefixes of the language. With this in mind, it follows that the DFA of $\Sigma^* = L(\Sigma, \Sigma^k)$ has states which correspond to the subsequences up to length $k-1$ of the prefixes of Σ^*. Figure 6 illustrates such a DFA when $k = 2$ and $\Sigma = \{a, b, c\}$.

In fact, these DFAs reveal the differences between SP languages and PT languages: they are exactly those expressed in Lemma 2. Within the state space defined by the subsequences up to length $k-1$ of the prefixes of the language, if the conditions in Lemma 2 are violated, then the DFAs describe languages that are PT but not SP. Pictorially, PT_2 languages are obtained by arbitrarily removing arcs, states, and the finality of states from the DFA in Figure 6, and SP_2 ones are obtained by non-arbitrarily removing them in accordance with Lemma 2. The same applies straightforwardly for any k (see Definition 3 below).

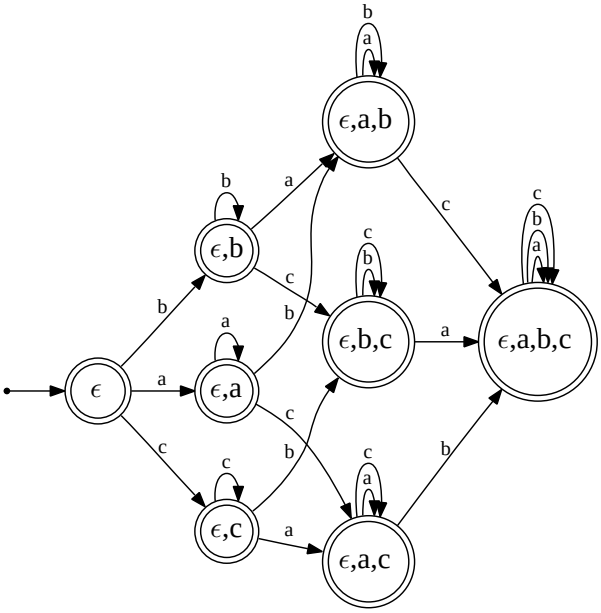


Figure 6: A DFA representation of the of the SP_2 language given by $\mathcal{G} = \langle\{a, b, c\}, \Sigma^2\rangle$. Names of the states reflect subsets of subsequences up to length 1 of prefixes of the language. Note this DFA is trimmed, but not minimal.

5 SP Distributions

In the same way that SL distributions (n-gram models) generalize SL languages, SP distributions generalize SP languages. Recall that SP languages are characterizable by the intersection of the complements of principal shuffle ideals. SP distributions are similarly characterized.

We begin with Piecewise-Testable distributions.

Definition 3 *A distribution $\mathcal{D}$ is k-Piecewise Testable (written $\mathcal{D} \in \text{PTD}_k$) $\overset{def}{\iff}$ $\mathcal{D}$ can be described by a PDFA $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F, T\rangle$ with*

1. $Q = \{P_{\leq k-1}(w) : w \in \Sigma^*\}$
2. $q_0 = P_{\leq k-1}(\epsilon)$
3. *For all $w \in \Sigma^*$ and all $\sigma \in \Sigma$, $\delta(P_{\leq k-1}(w), a) = P_{\leq k-1}(wa)$*
4. *F and T satisfy Equation 1.*

In other words, a distribution is k-Piecewise Testable provided it can be represented by a PDFA whose structural components are the same (modulo renaming of states) as those of the DFA discussed earlier where states corresponded to the subsequences up to length $k-1$ of the prefixes of the language. The DFA in Figure 6 shows the

structure of a PDFA which describes a PT_2 distribution as long as the assigned probabilities satisfy Equation 1.

The following lemma follows directly from the finite-state representation of PT_k distributions.

Lemma 3 *Let $\mathcal{D}$ belong to PTD_k and let $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F, T\rangle$ be a PDFA representing $\mathcal{D}$ defined according to Definition 3.*

$$Pr_D(\sigma_1 \dots \sigma_n) = T(P_{\leq k-1}(\epsilon), \sigma_1) \cdot \left(\prod_{2 \leq i \leq n} T(P_{\leq k-1}(\sigma_1 \dots \sigma_{i-1}), \sigma_i) \right) \cdot F(P_{\leq k-1}(w)) \quad (4)$$

PT_k distributions have $2^{|\Sigma|^{k-1}}(|\Sigma|+1)$ parameters (since there are $2^{|\Sigma|^{k-1}}$ states and $|\Sigma|+1$ possible events, i.e. transitions and finality).

Let $Pr(\sigma \mid \#)$ and $Pr(\# \mid P_{\leq k}(w))$ denote the probability (according to some $\mathcal{D} \in \text{PTD}_k$) that a word begins with σ and ends after observing $P_{\leq k}(w)$. Then Equation 4 can be rewritten in terms of conditional probability as

$$Pr_D(\sigma_1 \dots \sigma_n) = Pr(\sigma_1 \mid \#) \cdot \left(\prod_{2 \leq i \leq n} Pr(\sigma_i \mid P_{\leq k-1}(\sigma_1 \dots \sigma_{i-1})) \right) \cdot Pr(\# \mid P_{\leq k-1}(w)) \quad (5)$$

Thus, the probability assigned to a word depends not on the observed contiguous sequences as in a Markov model, but on observed subsequences.

Like SP languages, SP distributions can be defined in terms of the product of machines very similar to the complement of principal shuffle ideals.

Definition 4 *Let $w \in \Sigma^{k-1}$ and $w = \sigma_1 \cdots \sigma_{k-1}$. $\mathcal{M}_w = \langle Q, \Sigma, q_0, \delta, F, T\rangle$ is a* w-subsequence-distinguishing PDFA *(w-SD-PDFA) iff $Q = \text{Pfx}(w)$, $q_0 = \epsilon$, for all $u \in \text{Pfx}(w)$ and each $\sigma \in \Sigma$,*

$$\delta(u, \sigma) = \begin{array}{l} u\sigma \text{ iff } u\sigma \in \text{Pfx}(w) \text{ and} \\ u \text{ otherwise} \end{array}$$

and F and T satisfy Equation 1.

Figure 7 shows the structure of $\mathcal{M}_a$ which is almost the same as the complement of the principal shuffle ideal in Figure 4. The only difference is the additional self-loop labeled a on the rightmost state labeled a. $\mathcal{M}_a$ defines a family of distributions over Σ^*, and its states distinguish those

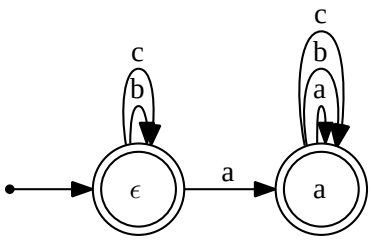


Figure 7: The structure of PDFA $\mathcal{M}_a$. It is the same (modulo state names) as the DFA in Figure 4 except for the self-loop labeled a on state a.

strings which contain a (state a) from those that do not (state ϵ). A set of PDFAs is *a k-set of SD-PDFAs* iff, for each $w \in \Sigma^{\leq k-1}$, it contains exactly one w-SD-PDFA.

In the same way that missing edges propagate down in DFA representations of SP languages (Lemma 2), the final and transitional probabilities must propagate down in PDFA representations of SP_k distributions. In other words, the final and transitional probabilities at states further along paths beginning at the start state must be determined by final and transitional probabilities at earlier states non-increasingly. This is captured by defining SP distributions as a product of k-sets of SD-PDFAs (see Definition 5 below).

While the standard product based on co-emission probability could be used for this purpose, we adopt a modified version of it defined for k-sets of SD-PDFAs: the *positive co-emission probability*. The automata product based on the positive co-emission probability not only ensures that the probabilities propagate as necessary, but also that such probabilities are made on the basis of observed subsequences, and not unobserved ones. This idea is familiar from n-gram models: the probability of σ_n given the immediately preceding sequence $\sigma_1 \dots \sigma_{n-1}$ does not depend on the probability of σ_n given the other $(n-1)$-long sequences which do not immediately precede it, though this is a logical possibility.

Let $\mathcal{A}$ be a k-set of SD-PDFAs. For each $w \in \Sigma^{\leq k-1}$, let $\mathcal{M}_w = \langle Q_w, \Sigma, q_{0w}, \delta_w, F_w, T_w\rangle$ be the w-subsequence-distinguishing PDFA in $\mathcal{A}$. The positive co-emission probability that σ is simultaneously emitted from states $q_\epsilon, \dots, q_u$ from the statesets $Q_\epsilon, \dots Q_u$, respectively, of each SD-

PDFA in $\mathcal{A}$ is

$$PCT(\langle \sigma, q_\epsilon \ \ldots \ q_u \rangle) = \prod_{\substack{q_w \in \langle q_\epsilon \ldots q_u \rangle \\ q_w = w}} T_w(q_w, \sigma) \quad (6)$$

Similarly, the probability that a word simultaneously ends at n states $q_\epsilon \in Q_\epsilon, \ldots, \ q_u \in Q_u$ is

$$PCF(\langle q_\epsilon \ \ldots \ q_u \rangle) = \prod_{\substack{q_w \in \langle q_\epsilon \ldots q_u \rangle \\ q_w = w}} F_w(q_w) \quad (7)$$

In other words, the positive co-emission probability is the product of the probabilities restricted to those assigned to the maximal states in each $\mathcal{M}_w$. For example, consider a 2-set of SD-PDFAs $\mathcal{A}$ with $\Sigma = \{a, b, c\}$. $\mathcal{A}$ contains four PDFAs $\mathcal{M}_\epsilon, \mathcal{M}_a, \mathcal{M}_b, \mathcal{M}_c$. Consider state $q = \langle \epsilon, \epsilon, b, c \rangle \in \bigotimes \mathcal{A}$ (this is the state labeled ϵ, b, c in Figure 6). Then

$$CT(a, q) = T_\epsilon(\epsilon, a) \cdot \ T_a(\epsilon, a) \cdot \ T_b(b, a) \cdot \ T_c(c, a)$$

but

$$PCT(a, q) = T_\epsilon(\epsilon, a) \cdot \ T_b(b, a) \cdot \ T_c(c, a)$$

since in PDFA $\mathcal{M}_a$, the state ϵ is not the maximal state.

The positive co-emission product ($\otimes^+$) is defined just as with co-emission probabilities, substituting PCT and PCF for CT and CF, respectively, in Definition 1. The definition of $\otimes^+$ ensures that the probabilities propagate on the basis of observed subsequences, and not on the basis of unobserved ones.

Lemma 4 *Let $k \geq 1$ and let $\mathcal{A}$ be a k-set of SD-PDFAs. Then $\otimes^+ S$ defines a well-formed probability distribution over Σ^*.*

Proof Since $\mathcal{M}_\epsilon$ belongs to $\mathcal{A}$, it is always the case that PCT and PCF are defined. Well-formedness follows from the normalization term as in Definition 1. ⊣

Definition 5 *A distribution $\mathcal{D}$ is k-Strictly Piecewise (written $\mathcal{D} \in \text{SPD}_k$) $\stackrel{def}{\Longleftrightarrow}$ $\mathcal{D}$ can be described by a PDFA which is the positive co-emission product of a k-set of subsequence-distinguishing PDFAs.*

By Lemma 4, SP distributions are well-formed. Unlike PDFAs for PT distributions, which distinguish $2^{|\Sigma|^{k-1}}$ states, the number of states in a k-set of SD-PDFAs is $\sum_{i<k}(i+1)|\Sigma|^i$, which is $\Theta(|\Sigma|^{k+1})$. Furthermore, since each SD-PDFA only has one state contributing $|\Sigma|+1$ probabilities to the product, and since there are $|\Sigma^{\leq k}| = \frac{|\Sigma|^k - 1}{|\Sigma| - 1}$ many SD-PDFAs in a k-set, there are

$$\frac{|\Sigma|^k - 1}{|\Sigma| - 1} \cdot (|\Sigma| + 1) = \frac{|\Sigma|^{k+1} + |\Sigma|^k - |\Sigma| - 1}{|\Sigma| - 1}$$

parameters, which is $\Theta(|\Sigma|^k)$.

Lemma 5 *Let $\mathcal{D} \in \text{SPD}_k$. Then $\mathcal{D} \in \text{PTD}_k$.*

Proof Since $\mathcal{D} \in \text{SPD}_k$, there is a k-set of subsequence-distinguishing PDFAs. The product of this set has the same structure as the PDFA given in Definition 3. ⊣

Theorem 2 *A distribution $\mathcal{D} \in \text{SPD}_k$ if $\mathcal{D}$ can be described by a PDFA $\mathcal{M} = \langle Q, \Sigma, q_0, \delta, F, T \rangle$ satisfying Definition 3 and the following.*

For all $w \in \Sigma^*$ and all $\sigma \in \Sigma$, let

$$Z(w) = \prod_{s \in P_{\leq k-1}(w)} F(P_{\leq k-1}(s)) + \sum_{\sigma' \in \Sigma} \left(\prod_{s \in P_{\leq k-1}(w)} T(P_{\leq k-1}(s), \sigma') \right) \quad (8)$$

(This is the normalization term.) Then T must satisfy: $T(P_{\leq k-1}(w), \sigma) =$

$$\frac{\prod_{s \in P_{\leq k-1}(w)} T(P_{\leq k-1}(s), \sigma)}{Z(w)} \quad (9)$$

and F must satisfy: $F(P_{\leq k-1}(w)) =$

$$\frac{\prod_{s \in P_{\leq k-1}(w)} F(P_{\leq k-1}(s))}{Z(w)} \quad (10)$$

Proof That SPD_k satisfies Definition 3 Follows directly from Lemma 5. Equations 8-10 follow from the definition of positive co-emission probability. ⊣

The way in which final and transitional probabilities propagate down in SP distributions is reflected in the conditional probability as defined by Equations 9 and 10. In terms of conditional probability, Equations 9 and 10 mean that the probability that σ_i follows a sequence $\sigma_1 \ldots \sigma_{i-1}$ is not only a function of $P_{\leq k-1}(\sigma_1 \ldots \sigma_{i-1})$ (Equation 4) but further that it is a function of each subsequence in $\sigma_1 \ldots \sigma_{i-1}$ up to length $k-1$.

In particular, $Pr(\sigma_i \mid P_{\leq k-1}(\sigma_1 \ldots \sigma_{i-1}))$ is obtained by substituting $Pr(\sigma_i \mid P_{\leq\, k-1}(s))$ for $T(P_{\leq\, k-1}(s), \sigma)$ and $Pr(\# \mid P_{\leq\, k-1}(s))$ for $F(P_{\leq k-1}(s))$ in Equations 8, 9 and 10. For example, for a SP_2 distribution, the probability of a given $P_{\leq 1}(bc)$ (state ϵ, b, c in Figure 6) is the normalized product of the probabilities of a given $P_{\leq 1}(\epsilon)$, a given $P_{\leq 1}(b)$, and a given $P_{\leq 1}(c)$.

To summarize, SP and PT distributions are regular deterministic. Unlike PT distributions, however, SP distributions can be modeled with only $\Theta(|\Sigma|^k)$ parameters and $\Theta(|\Sigma|^{k+1})$ states. This is true even though SP distributions distinguish $2^{|\Sigma|^{k-1}}$ states! Since SP distributions can be represented by a single PDFA, computing $Pr(w)$ occurs in only $\Theta(|w|)$ for such PDFA. While such PDFA might be too large to be practical, $Pr(w)$ can also be computed from the k-set of SD-PDFAs in $\Theta(|w|^k)$ (essentially building the path in the product machine on the fly using Equations 4, 8, 9 and 10).

6 Estimating SP Distributions

The problem of ML estimation of SP_k distributions is reduced to estimating the parameters of the SD-PDFAs. Training (counting and normalization) occurs over each of these machines (i.e. each machine parses the entire corpus), which gives the ML estimates of the parameters of the distribution. It trivially follows that this training successfully estimates any $\mathcal{D} \in \text{SPD}_k$.

Theorem 3 *For any $\mathcal{D} \in \text{SPD}_k$, let $\mathcal{D}$ generate sample S. Let $\mathcal{A}$ be the k-set of SD-PDFAs which describes exactly $\mathcal{D}$. Then optimizing the MLE of S with respect to each $\mathcal{M} \in \mathcal{A}$ guarantees that the distribution described by the positive co-emission product of $\bigotimes^+ \mathcal{A}$ approaches $\mathcal{D}$ as $|S|$ increases.*

Proof The MLE estimate of S with respect to SPD_k returns the parameter values that maximize the likelihood of S. The parameters of $\mathcal{D} \in \text{SPD}_k$ are found on the maximal states of each $\mathcal{M} \in \mathcal{A}$. By definition, each $\mathcal{M} \in \mathcal{A}$ describes a probability distribution over Σ^*, and similarly defines a family of distributions. Therefore finding the MLE of S with respect to SPD_k means finding the MLE estimate of S with respect to each of the family of distributions which each $\mathcal{M} \in \mathcal{A}$ defines, respectively.

Optimizing the ML estimate of S for each $\mathcal{M} \in \mathcal{A}$ means that as $|S|$ increases, the estimates $\hat{T}_{\mathcal{M}}$ and $\hat{F}_{\mathcal{M}}$ approach the true values $T_{\mathcal{M}}$ and $F_{\mathcal{M}}$. It follows that as $|S|$ increases, $\hat{T}_{\bigotimes^+ \mathcal{A}}$ and $\hat{F}_{\bigotimes^+ \mathcal{A}}$ approach the true values of $T_{\bigotimes^+ \mathcal{A}}$ and $F_{\bigotimes^+ \mathcal{A}}$ and consequently $\mathcal{D}_{\bigotimes^+ \mathcal{A}}$ approaches $\mathcal{D}$. ⊣

We demonstrate learning long-distance dependencies by estimating SP_2 distributions given a corpus from Samala (Chumash), a language with sibilant harmony.[4] There are two classes of sibilants in Samala: [-anterior] sibilants like [s] and [t͡s] and [+anterior] sibilants like [ʃ] and [t͡ʃ].[5] Samala words are subject to a phonological process wherein the last sibilant requires earlier sibilants to have the same value for the feature [anterior], no matter how many sounds intervene (Applegate, 1972). As a consequence of this rule, there are generally no words in Samala where [-anterior] sibilants follow [+anterior]. E.g. [ʃtojonowonowaʃ] 'it stood upright' (Applegate 1972:72) is licit but not *[ʃtojonowonowas].

The results of estimating $\mathcal{D} \in \text{SPD}_2$ with the corpus is shown in Table 6. The results clearly demonstrate the effectiveness of the model: the probability of a [α anterior] sibilant given $P_{\leq 1}$([-α anterior]) sounds is orders of magnitude less than given $P_{\leq 1}(\alpha$ anterior]) sounds.

$Pr(x \mid P_{\leq 1}(y))$		x			
		s	t͡s	ʃ	t͡ʃ
y	s	0.0335	0.0051	0.0011	0.0002
	t͡s	0.0218	0.0113	0.0009	0.
	ʃ	0.0009	0.	0.0671	0.0353
	t͡ʃ	0.0006	0.	0.0455	0.0313

Table 1: Results of SP_2 estimation on the Samala corpus. Only sibilants are shown.

7 Conclusion

SP distributions are the stochastic version of SP languages, which model long-distance dependencies. Although SP distributions distinguish $2^{|\Sigma|^{k-1}}$ states, they do so with tractably many parameters and states because of an assumption that distinct subsequences do not interact. As shown, these distributions are efficiently estimable from positive data. As previously mentioned, we anticipate these models to find wide application in NLP.

[4] The corpus was kindly provided by Dr. Richard Applegate and drawn from his 2007 dictionary of Samala.

[5] Samala actually contrasts glottalized, aspirated, and plain variants of these sounds (Applegate, 1972). These laryngeal distinctions are collapsed here for easier exposition.

References

R.B. Applegate. 1972. *Ineseño Chumash Grammar*. Ph.D. thesis, University of California, Berkeley.

R.B. Applegate. 2007. *Samala-English dictionary : a guide to the Samala language of the Ineseño Chumash People*. Santa Ynez Band of Chumash Indians.

Eric Baković. 2000. *Harmony, Dominance and Control*. Ph.D. thesis, Rutgers University.

D. Beauquier and Jean-Eric Pin. 1991. Languages and scanners. *Theoretical Computer Science*, 84:3–21.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–566.

J. A. Brzozowski and Imre Simon. 1973. Characterizations of locally testable events. *Discrete Mathematics*, 4:243–271.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*. IT-2.

J. S. Coleman and J. Pierrehumbert. 1997. Stochastic phonological grammars and acceptability. In *Computational Phonology*, pages 49–56. Somerset, NJ: Association for Computational Linguistics. Third Meeting of the ACL Special Interest Group in Computational Phonology.

Colin de la Higuera. in press. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Pedro García and José Ruiz. 1990. Inference of k-testable languages in the strict sense and applications to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:920–925.

Pedro García and José Ruiz. 1996. Learning k-piecewise testable languages from positive data. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Interference: Learning Syntax from Sentences*, volume 1147 of *Lecture Notes in Computer Science*, pages 203–210. Springer.

Pedro Garcia, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338.

Gunnar Hansson. 2001. *Theoretical and typological issues in consonant harmony*. Ph.D. thesis, University of California, Berkeley.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39:379–440.

Jeffrey Heinz. 2007. *The Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles.

Jeffrey Heinz. to appear. Learning long distance phonotactics. *Linguistic Inquiry*.

John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Frederick Jelenik. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

C. Douglas Johnson. 1972. *Formal Aspects of Phonological Description*. The Hague: Mouton.

A. K. Joshi. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2nd edition.

Ronald Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Gregory Kobele. 2006. *Generating Copies: An Investigation into Structural Identity in Language and Grammar*. Ph.D. thesis, University of California, Los Angeles.

Leonid (Aryeh) Kontorovich, Corinna Cortes, and Mehryar Mohri. 2008. Kernel methods for learning languages. *Theoretical Computer Science*, 405(3):223 – 236. Algorithmic Learning Theory.

M. Lothaire, editor. 1997. *Combinatorics on Words*. Cambridge University Press, Cambridge, UK, New York.

A. A. Markov. 1913. An example of statistical study on the text of 'eugene onegin' illustrating the linking of events to a chain.

Robert McNaughton and Simon Papert. 1971. *Counter-Free Automata*. MIT Press.

A. Newell, S. Langer, and M. Hickey. 1998. The rôle of natural language processing in alternative and augmentative communication. *Natural Language Engineering*, 4(1):1–16.

Dominique Perrin and Jean-Eric Pin. 1986. First-Order logic and Star-Free sets. *Journal of Computer and System Sciences*, 32:393–406.

Catherine Ringen. 1988. *Vowel Harmony: Theoretical Implications*. Garland Publishing, Inc.

James Rogers and Geoffrey Pullum. to appear. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*.

James Rogers, Jeffrey Heinz, Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. to appear. On languages piecewise testable in the strict sense. In *Proceedings of the 11th Meeting of the Assocation for Mathematics of Language*.

Sharon Rose and Rachel Walker. 2004. A typology of consonant agreement as correspondence. *Language*, 80(3):475–531.

Jacques Sakarovitch and Imre Simon. 1983. Subwords. In M. Lothaire, editor, *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*, chapter 6, pages 105–134. Addison-Wesley, Reading, Massachusetts.

Stuart Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages: 2nd Grammatical Inference conference*, pages 214–222, Berlin ; New York. Springer-Verlag.

Howard Straubing. 1994. *Finite Automata, Formal Logic and Circuit Complexity*. Birkhäuser.

Wolfgang Thomas. 1982. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences*, 25:360–376.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005a. Probabilistic finite-state machines-part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.

Enrique Vidal, Frank Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005b. Probabilistic finite-state machines-part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039.

String Extension Learning

Jeffrey Heinz
University of Delaware
Newark, Delaware, USA
heinz@udel.edu

Abstract

This paper provides a unified, learning-theoretic analysis of several learnable classes of languages discussed previously in the literature. The analysis shows that for these classes an incremental, globally consistent, locally conservative, set-driven learner always exists. Additionally, the analysis provides a recipe for constructing new learnable classes. Potential applications include learnable models for aspects of natural language and cognition.

1 Introduction

The problem of generalizing from examples to patterns is an important one in linguistics and computer science. This paper shows that many disparate language classes, many previously discussed in the literature, have a simple, natural and interesting (because non-enumerative) learner which exactly identifies the class in the limit from distribution-free, positive evidence in the sense of Gold (Gold, 1967).[1] These learners are called String Extension Learners because each string in the language can be mapped (extended) to an element of the grammar, which in every case, is conceived as a finite set of elements. These learners have desirable properties: they are incremental, globally consistent, and locally conservative.

Classes previously discussed in the literature which are string extension learnable include the Locally Testable (LT) languages, the Locally Testable Languages in the Strict Sense (Strictly Local, SL) (McNaughton and Papert, 1971; Rogers and Pullum, to appear), the Piecewise Testable (PT) languages (Simon, 1975), the Piecewise Testable languages in the Strict Sense (Strictly Piecewise, SP) (Rogers et al., 2009), the Strongly Testable languages (Beauquier and Pin, 1991), the Definite languages (Brzozowski, 1962), and the Finite languages, among others. To our knowledge, this is the first analysis which identifies the common structural elements of these language classes which allows them to be identifiable in the limit from positive data: each language class induces a natural partition over all logically possible strings and each language in the class is the union of finitely many blocks of this partition.

One consequence of this analysis is a recipe for constructing new learnable classes. One notable case is the Strictly Piecewise (SP) languages, which was originally motivated for two reasons: the learnability properties discussed here and its ability to describe long-distance dependencies in natural language phonology (Heinz, 2007; Heinz, to appear). Later this class was discovered to have several independent characterizations and form the basis of another subregular hierarchy (Rogers et al., 2009).

It is expected string extension learning will have applications in linguistic and cognitive models. As mentioned, the SP languages already provide a novel hypothesis of how long-distance dependencies in sound patterns are learned. Another example is the Strictly Local (SL) languages which are the categorical, symbolic version of n-gram models, which are widely used in natural language processing (Jurafsky and Martin, 2008). Since the SP languages also admit a probabilistic variant which describe an efficiently estimable class of distributions (Heinz and Rogers, 2010), it is plausible to expect the other classes will as well, though this is left for future research.

String extension learners are also simple, mak-

[1] The allowance of negative evidence (Gold, 1967) or restricting the kinds of texts the learner is required to succeed on (i.e. non-distribution-free evidence) (Gold, 1967; Horning, 1969; Angluin, 1988) admits the learnability of the class of recursively enumerable languages. Classes of languages learnable in the harder, distribution-free, positive-evidence-only settings are due to structural properties of the language classes that permit generalization (Angluin, 1980b; Blumer et al., 1989). That is the central interest here.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 897–906,
Uppsala, Sweden, 11-16 July 2010.

ing them accessible to linguists without a rigorous mathematical background.

This paper is organized as follow. §2 goes over basic notation and definitions. §3 defines string extension grammars, languages, and language classes and proves some of their fundamental properties. §4 defines string extension learners and proves their behavior. §5 shows how important subregular classes are string extension language classes. §6 gives examples of nonregular and infinite language classes which are string extension learnable. §7 summarizes the results, and discusses lines of inquiry for future research.

2 Preliminaries

This section establishes notation and recalls basic definitions for formal languages, the paradigm of identification in the limit from positive data (Gold, 1967). Familiarity with the basic concepts of sets, functions, and sequences is assumed.

For some set A, $\mathcal{P}(A)$ denotes the set of all subsets of A and $\mathcal{P}_{fin}(A)$ denotes the set of all finite subsets of A. If f is a function such that $f : A \rightarrow B$ then let $f^{\diamond}(a) = \{f(a)\}$. Thus, $f^{\diamond} : A \rightarrow \mathcal{P}(B)$ (note $f^{\diamond}$ is not surjective). A set π of nonempty subsets of S is a *partition* of S iff the elements of π (called *blocks*) are pairwise disjoint and their union equals S.

Σ denotes a fixed finite set of symbols, the *alphabet*. Let Σ^n, $\Sigma^{\leq n}$, Σ^*, Σ^+ denote all strings formed over this alphabet of length n, of length less than or equal to n, of any finite length, and of any finite length strictly greater than zero, respectively. The term *word* is used interchangeably with *string*. The *range* of a string w is the set of symbols which are in w. The empty string is the unique string of length zero denoted λ. Thus $range(\lambda) = \emptyset$. The length of a string u is denoted by $|u|$, e.g. $|\lambda| = 0$. A language L is some subset of Σ^*. The reverse of a language $L^r = \{w^r : w \in L\}$.

Gold (1967) establishes a learning paradigm known as identification in the limit from positive data. A *text* is an infinite sequence whose elements are drawn from $\Sigma^* \cup \{\#\}$ where $\#$ represents a non-expression. The ith element of t is denoted $t(i)$, and $t[i]$ denotes the finite sequence $t(0), t(1), \ldots t(i)$. Following Jain et al. (1999), let SEQ denote the set of all possible finite sequences:

$$SEQ = \{t[i] : t \text{ is a text and } i \in \mathbb{N}\}$$

The *content* of a text is defined below.

$$content(t) = \{w \in \Sigma^* : \exists n \in \mathbb{N} \text{ such that } t(n) = w\}$$

A text t is a *positive text for a language* L iff $content(t) = L$. Thus there is only one text t for the empty language: for all i, $t(i) = \#$.

A *learner* is a function ϕ which maps initial finite sequences of texts to grammars, i.e. $\phi : SEQ \rightarrow \mathcal{G}$. The elements of $\mathcal{G}$ (the grammars) generate languages in some well-defined way. A learner *converges on a text* t iff there exists $i \in \mathbb{N}$ and a grammar G such that for all $j > i$, $\phi(t[j]) = G$.

For any grammar G, the language it generates is denoted $L(G)$. A learner ϕ *identifies a language* L *in the limit* iff for any positive text t for L, ϕ converges on t to grammar G and $L(G) = L$. Finally, a learner ϕ *identifies a class of languages* $\mathcal{L}$ *in the limit* iff for any $L \in \mathcal{L}$, ϕ identifies L in the limit. Angluin (1980b) provides necessary and sufficient properties of language classes which are identifiable in the limit from positive data.

A learner ϕ of language class $\mathcal{L}$ is *globally consistent* iff for each i and for all texts t for some $L \in \mathcal{L}$, $content(t[i]) \subseteq L(\phi(t[i]))$. A learner ϕ is *locally conservative* iff for each i and for all texts t for some $L \in \mathcal{L}$, whenever $\phi(t[i]) \neq \phi(t[i-1])$, it is the case that $t(i) \notin L(\phi([i-1]))$. These terms are from Jain et al. (2007). Also, learners which do not depend on the order of the text are called *set-driven* (Jain et al., 1999, p. 99).

3 Grammars and Languages

Consider some set A. A string extension function is a total function $f : \Sigma^* \rightarrow \mathcal{P}_{fin}(A)$. It is not required that f be onto. Denote the class of functions which have this general form $\mathcal{SEF}$.

Each string extension function is naturally associated with some formal class of grammars and languages. These functions, grammars, and languages are called *string extension functions*, *grammars*, and *languages*, respectively.

Definition 1 Let $f \in \mathcal{SEF}$.

1. A *grammar* is a finite subset of A.
2. The *language of grammar* G is

$$L_f(G) = \{w \in \Sigma^* : f(w) \subseteq G\}$$

3. The *class of languages* obtained by all possible grammars is

$$\mathcal{L}_f = \{L_f(G) : G \in \mathcal{P}_{fin}(A)\}$$

The subscript f is omitted when it is understood from context.

A function $f \in \mathcal{SEF}$ naturally induces a partition π_f over Σ^*. Strings u and v are equivalent ($u \sim_f v$) iff $f(u) = f(v)$.

Theorem 1 Every language $L \in \mathcal{L}_f$ is a finite union of blocks of π_f.

Proof: Follows directly from the definition of $\sim_f$ and the finiteness of string extension grammars. □

We return to this result in §6.

Theorem 2 $\mathcal{L}_f$ is closed under intersection.

Proof: We show $L_1 \cap L_2 = L(G_1 \cap G_2)$. Consider any word w belonging to L_1 and L_2. Then $f(w)$ is a subset of G_1 and of G_2. Thus $f(w) \subseteq G_1 \cap G_2$, and therefore $w \in L(G_1 \cap G_2)$. The other inclusion follows similarly. □

String extension language classes are not in general closed under union or reversal (counterexamples to union closure are given in §5.1 and to reversal closure in §6.)

It is useful to extend the domain of the function f from strings to languages.

$$f(L) = \bigcup_{w \in L} f(w) \tag{1}$$

An element g of grammar G for language $L = L_f(G)$ is *useful* iff $g \in f(L)$. An element is *useless* if it is not useful. A grammar with no useless elements is called *canonical*.

Remark 1 *Fix a function $f \in \mathcal{SEF}$. For every $L \in \mathcal{L}_f$, there is a canonical grammar, namely $f(L)$. In other words, $L = L(f(L))$.*

Lemma 1 Let $L, L' \in \mathcal{L}_f$. $L \subseteq L'$ iff $f(L) \subseteq f(L')$

Proof: ($\Rightarrow$) Suppose $L \subseteq L'$ and consider any $g \in f(L)$. Since g is useful, there is a $w \in L$ such that $g \in f(w)$. But $f(w) \subseteq f(L')$ since $w \in L'$.

($\Leftarrow$) Suppose $f(L) \subseteq f(L')$ and consider any $w \in L$. Then $f(w) \subseteq f(L)$ so by transitivity, $f(w) \subseteq f(L')$. Therefore $w \in L'$. □

The significance of this result is that as the grammar G monotonically increases, the language $L(G)$ monotonically increases too. The following result can now be proved, used in the next section on learning.[2]

Theorem 3 For any finite $L_0 \subseteq \Sigma^*$, $L = L(f(L_0))$ is the smallest language in $\mathcal{L}_f$ containing L_0.

Proof: Clearly $L_0 \subseteq L$. Suppose $L' \in \mathcal{L}_f$ and $L_0 \subseteq L'$. It follows directly from Lemma 1 that $L \subseteq L'$ (since $f(L) = f(L_0) \subseteq f(L')$). □

4 String Extension Learning

Learning string extension classes is simple. The initial hypothesis of the learner is the empty grammar. The learner's next hypothesis is obtained by applying function f to the current observation and taking the union of that set with the previous one.

Definition 2 For all $f \in \mathcal{SEF}$ and for all $t \in SEQ$, define ϕ_f as follows:

$$\phi_f(t[i]) = \begin{cases} \emptyset & \text{if } i = -1 \\ \phi_f(t[i-1]) & \text{if } t(i) = \# \\ \phi_f(t[i-1]) \cup f(t(i)) & \text{otherwise} \end{cases}$$

By convention, the initial state of the grammar is given by $\phi(t[-1]) = \emptyset$. The learner ϕ_f exemplifies *string extension learning*. Each individual string in the text reveals, by extension with f, aspects of the canonical grammar for $L \in \mathcal{L}_f$.

Theorem 4 ϕ_f is globally consistent, locally conservative, and set-driven.

Proof: Global consistness and local conservativeness follow immediately from Definition 2. For set-drivenness, witness (by Definition 2) it is the case that for any text t and any $i \in \mathbb{N}$, $\phi(t[i]) = f(content(t[i]))$. □

The key to the proof that ϕ_f identifies $\mathcal{L}_f$ in the limit from positive data is the finiteness of G for all $L(G) \in \mathcal{L}$. The idea is that there is a point in the text in which every element of the grammar has been seen because (1) there are only finitely many useful elements of G, and (2) the learner is guaranteed to see a word in L which yields (via f) each element of G at some point (since the learner receives a positive text for L). Thus at this point

[2]The requirement in Theorem 3 that L_0 be finite can be dropped if the qualifier "in $\mathcal{L}_f$" be dropped as well. This can be seen when one considers the identity function and the class of finite languages. (The identity function is a string extension function, see §6.) In this case, $id(\Sigma^*) = \Sigma^*$, but Σ^* is not a member of $\mathcal{L}_{fin}$. However since the interest here is learners which generalize on the basis of finite experience, Theorem 3 is sufficient as is.

the learner ϕ is guaranteed to have converged to the target G as no additional words will add any more elements to the learner's grammar.

Lemma 2 For all $L \in \mathcal{L}_f$, there is a finite sample S such that L is the smallest language in $\mathcal{L}_f$ containing S. S is called a *characteristic sample* of L in $\mathcal{L}_f$ (S is also called a *tell-tale*).

Proof: For $L \in \mathcal{L}_f$, construct the sample S as follows. For each $g \in f(L)$, choose some word $w \in L$ such that $g \in f(w)$. Since $f(L)$ is finite (Remark 1), S is finite. Clearly $f(S) = f(L)$ and thus $L = L(f(S))$. Therefore, by Theorem 3, L is the smallest language in $\mathcal{L}_f$ containing S. □

Theorem 5 Fix $f \in \mathcal{SEF}$. Then ϕ_f identifies $\mathcal{L}_f$ in the limit.

Proof: For any $L \in \mathcal{L}_f$, there is a characteristic finite sample S for L (Lemma 2). Thus for any text t for L, there is i such that $S \subseteq content(t[i])$. Thus for any $j > i$, $\phi(t(j))$ is the smallest language in $\mathcal{L}_f$ containing S by Theorem 3 and Lemma 2. Thus, $\phi(t(j)) = f(S) = f(L)$. □

An immediate corollary is the efficiency of ϕ_f in the length of the sample, provided f is efficient in the length of the string (de la Higuera, 1997).

Corollary 1 ϕ_f is efficient in the length of the sample iff f is efficiently computable in the length of a string.

To summarize: string extension grammars are finite subsets of some set A. The class of languages they generate are determined by a function f which maps strings to finite subsets of A (chunks of grammars). Since the size of the canonical grammars is finite, a learner which develops a grammar on the basis of the observed words and the function f identifies this class exactly in the limit from positive data. It also follows that if f is efficient in the length of the string then ϕ_f is efficient in the length of the sample and that ϕ_f is globally consistent, locally conservative, and set-driven. It is striking that such a natural and general framework for generalization exists and that, as will be shown, a variety of language classes can be expressed given the choice of f.

5 Subregular examples

This section shows how classes which make up the subregular hierarchies (McNaughton and Papert, 1971) are string extension language classes. Readers are referred to Rogers and Pullum (2007) and Rogers et al. (2009) for an introduction to the subregular hierarchies, as well as their relevance to linguistics and cognition.

5.1 K-factor languages

The k-factors of a word are the contiguous subsequences of length k in w. Consider the following string extension function.

Definition 3 For some $k \in \mathbb{N}$, let

$$fac_k(w) = \{x \in \Sigma^k : \exists u, v \in \Sigma^* \text{ such that } w = uxv\} \text{ when } k \leq |w| \text{ and } \{w\} \text{ otherwise}$$

Following the earlier definitions, for some k, a grammar G is a subset of $\Sigma^{\leq k}$ and a word w belongs to the language of G iff $fac_k(w) \subseteq G$.

Example 1 Let $\Sigma = \{a, b\}$ and consider grammars $G = \{\lambda, a, aa, ab, ba\}$. Then $L(G) = \{\lambda, a\} \cup \{w : |w| \geq 2 \text{ and } w \notin \Sigma^* bb \Sigma^*\}$. The 2-factor bb is a *prohibited* 2-factor for $L(G)$. Clearly, $L(G) \in \mathcal{L}_{fac_2}$.

Languages in $\mathcal{L}_{fac_k}$ make distinctions based on which k-factors are permitted or prohibited. Since $fac_k \in \mathcal{SEF}$, it follows immediately from the results in §§3-4 that the k-factor languages are closed under intersection, and each has a characteristic sample. For example, a characteristic sample for the 2-factor language in Example 1 is $\{\lambda, a, ab, ba, aa\}$; i.e. the canonical grammar itself. It follows from Theorem 5 that the class of k-factor languages is identifiable in the limit by ϕ_{fac_k}. The learner ϕ_{fac_2} with a text from the language in Example 1 is illustrated in Table 1.

The class $\mathcal{L}_{fac_k}$ is not closed under union. For example for $k = 2$, consider $L_1 = L(\{\lambda, a, b, aa, bb, ba\})$ and $L_2 = L(\{\lambda, a, b, aa, ab, bb\})$. Then $L_1 \cup L_2$ excludes string aba, but includes ab and ba, which is not possible for any $L \in \mathcal{L}_{fac_k}$.

K-factors are used to define other language classes, such as the Strictly Local and Locally Testable languages (McNaughton and Papert, 1971), discussed in §5.4 and §5.5.

5.2 Strictly k-Piecewise languages

The Strictly k-Piecewise (SP_k) languages (Rogers et al., 2009) can be defined with a function whose co-domain is $\mathcal{P}(\Sigma^{\leq k})$. However unlike the function fac_k, the function SP_k, does not require that the k-length subsequences be contiguous.

i	$t(i)$	$fac_2(t(i))$	Grammar G	$L(G)$
-1			$\emptyset$	$\emptyset$
0	$aaaa$	$\{aa\}$	$\{\mathbf{aa}\}$	aaa^*
1	aab	$\{aa,\ ab\}$	$\{aa,\ \mathbf{ab}\}$	$aaa^* \cup aaa^*b$
2	a	$\{a\}$	$\{\mathbf{a},\ aa,\ ab\}$	$aa^* \cup aa^*b$
...				

Table 1: The learner ϕ_{fac_2} with a text from the language in Example 1. Boldtype indicates newly added elements to the grammar.

A string $u = a_1 \dots a_k$ is a *subsequence* of string w iff $\exists\ v_0, v_1, \dots\ v_k \in \Sigma^*$ such that $w = v_0 a_1 v_1 \dots a_k v_k$. The empty string λ is a subsequence of every string. When u is a subsequence of w we write $u \sqsubseteq w$.

Definition 4 For some $k \in \mathbb{N}$,

$$SP_k(w) = \{u \in \Sigma^{\leq k} : u \sqsubseteq w\}$$

In other words, $SP_k(w)$ returns all subsequences, contiguous or not, in w up to length k. Thus, for some k, a grammar G is a subset of $\Sigma^{\leq k}$. Following Definition 1, a word w belongs to the language of G only if $SP_2(w) \subseteq G$.[3]

Example 2 Let $\Sigma = \{a, b\}$ and consider the grammar $G = \{\lambda, a, b, aa, ab, ba\}$. Then $L(G) = \Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$.

As seen from Example 2, SP languages encode long-distance dependencies. In Example 2, L prohibits a b from following another b in a word, no matter how distant. Table 2 illustrates ϕ_{SP_2} learning the language in Example 2.

Heinz (2007,2009a) shows that consonantal harmony patterns in natural language are describable by such SP_2 languages and hypothesizes that humans learn them in the way suggested by ϕ_{SP_2}. Strictly 2-Piecewise languages have also been used in models of reading comprehension (Whitney, 2001; Grainger and Whitney, 2004; Whitney and Cornelissen, 2008) as well as text classification(Lodhi et al., 2002; Cancedda et al., 2003) (see also (Shawe-Taylor and Christianini, 2005, chap. 11)).

5.3 K-Piecewise Testable languages

A language L is k-Piecewise Testable iff whenever strings u and v have the same subsequences of length at most k and u is in L, then v is in L as well (Simon, 1975; Simon, 1993; Lothaire, 2005).

A language L is said to be Piecewise-Testable (PT) if it is k-Piecewise Testable for some $k \in \mathbb{N}$. If k is fixed, the k-Piecewise Testable languages are identifiable in the limit from positive data (García and Ruiz, 1996; García and Ruiz, 2004). More recently, the Piecewise Testable languages has been shown to be linearly separable with a subsequence kernel (Kontorovich et al., 2008).

The k-Piecewise Testable languages can also be described with the function $SP_k^\diamond$. Recall that $f^\diamond(a) = \{f(a)\}$. Thus functions $SP_k^\diamond$ define grammars as a finite list of *sets of subsequences* up to length k that may occur in words in the language. This reflects the fact that the k-Piecewise Testable languages are the boolean closure of the Strictly k-Piecewise languages.[4]

5.4 Strictly k-Local languages

To define the Strictly k-Local languages, it is necessary to make a pointwise extension to the definitions in §3.

Definition 5 For sets $A_1, \dots, A_n$, suppose for each i, $f_i : \Sigma^* \to \mathcal{P}_{fin}(A_i)$, and let $f = (f_1, \dots, f_n)$.

1. A *grammar* G is a tuple $(G_1, \dots, G_n)$ where $G_1 \in \mathcal{P}_{fin}(A_1), \dots, G_n \in \mathcal{P}_{fin}(A_n)$.
2. If for any $w \in \Sigma^*$, each $f_i(w) \subseteq G_i$ for all $1 \leq i \leq n$, then $f(w)$ is a *pointwise subset* of G, written $f(w) \subseteq G$.
3. The *language of grammar* G is
$$L_f(G) = \{w : f(w) \subseteq G\}$$
4. The *class of languages* obtained by all such possible grammars G is $\mathcal{L}_f$.

[3] In earlier work, the function SP_2 has been described as returning the set of precedence relations in w, and the language class $\mathcal{L}_{SP_2}$ was called the precedence languages (Heinz, 2007; Heinz, to appear).

[4] More generally, it is not hard to show that $\mathcal{L}_{f^\diamond}$ is the boolean closure of $\mathcal{L}_f$.

i	$t(i)$	$SP_2(t(i))$	Grammar G	Language of G
-1			$\emptyset$	$\emptyset$
0	$aaaa$	$\{\lambda, a, aa\}$	$\{\lambda$**, a, aa**$\}$	a^*
1	aab	$\{\lambda, a, b, aa, ab\}$	$\{\lambda$, a, aa, **b, ab**$\}$	$a^* \cup a^*b$
2	baa	$\{\lambda, a, b, aa, ba\}$	$\{\lambda$, a, b, aa, ab, **ba**$\}$	$\Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$
3	aba	$\{\lambda, a, b, ab, ba\}$	$\{\lambda$, a, b, aa, ab, ba$\}$	$\Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$
...				

Table 2: The learner ϕ_{SP_2} with a text from the language in Example 2. Boldtype indicates newly added elements to the grammar.

These definitions preserve the learning results of §4. Note that the characteristic sample of $L \in \mathcal{L}_f$ will be the union of the characteristic samples of each f_i and the language $L_f(G)$ is the intersection of $L_{f_i}(G_i)$.

Locally k-Testable Languages in the Strict Sense (Strictly k-Local) have been studied by several researchers (McNaughton and Papert, 1971; Garcia et al., 1990; Caron, 2000; Rogers and Pullum, to appear), among others. We follow the definitions from (McNaughton and Papert, 1971, p. 14), effectively encoded in the following functions.

Definition 6 Fix $k \in \mathbb{N}$. Then the (left-edge) prefix of length k, the (right-edge) suffix of length k, and the interior k-factors of a word w are

$$L_k(w) = \{u \in \Sigma^k : \exists v \in \Sigma^* \text{ such that } w = uv\}$$

$$R_k(w) = \{u \in \Sigma^k : \exists v \in \Sigma^* \text{ such that } w = vu\}$$

$$I_k(w) = fac_k(w) \backslash (L_k(w) \cup R_k(w))$$

Example 3 Suppose $w = abcba$. Then $L_2(w) = \{ab\}$, $R_2(w) = \{ba\}$ and $I_2(w) = \{bc,\ cb\}$.

Example 4 Suppose $|w| = k$. Then $L_k(w) = R_k(w) = \{w\}$ and $I_k(w) = \emptyset$.

Example 5 Suppose $|w|$ is less than k. Then $L_k(w) = R_k(w) = \emptyset$ and $I_k(w) = \{w\}$.

A language L is *k-Strictly Local* (k-SL) iff for all $w \in L$, there exist sets $L, R,$ and I such that $w \in L$ iff $L_k(w) \subseteq L$, $R_k(w) \subseteq R$, and $I_k(w) \subseteq I$. McNaughton and Papert note that if w is of length less than k than L may be perfectly arbitrary about w.

This can now be expressed as the string extension function:

$$LRI_k(w) = (L_k(w), R_k(w), I_k(w))$$

Thus for some k, a grammar G is triple formed by taking subsets of Σ^k, Σ^k, and $\Sigma^{\leq k}$, respectively. A word w belongs to the language of G only if $LRI_k(w) \subseteq G$. Clearly, $\mathcal{L}_{LRI_k} = k$-SL, and henceforth we refer to this class as k-SL. Since, for fixed k, $LRI_k \in \mathcal{SEF}$, all of the learning results in §4 apply.

5.5 Locally k-Testable languages

The Locally k-testable languages (k-LT) are originally defined in McNaughton and Papert (1971) and are the subject of several studies (Brzozowski and Simon, 1973; McNaughton, 1974; Kim et al., 1991; Caron, 2000; García and Ruiz, 2004; Rogers and Pullum, to appear).

A language L is k-testable iff for all $w_1, w_2 \in \Sigma^*$ such that $|w_1| \geq k$ and $|w_2| \geq k$, and $LRI_k(w_1) = LRI_k(w_2)$ then either both w_1, w_2 belong to L or neither do. Clearly, every language in k-SL belongs to k-LT. However k-LT properly include k-SL because a k-testable language only distinguishes words whenever $LRI_k(w_1) \neq LRI_k(w_2)$. It is known that the k-LT languages are the boolean closure of the k-SL (McNaughton and Papert, 1971).

The function $LRI_k^\diamond$ exactly expresses k-testable languages. Informally, each word w is mapped to a set containing a single element, this element is the triple $LRI_k(w)$. Thus a grammar G is a subset of the triples used to define k-SL. Clearly, $\mathcal{L}_{LRI_k^\diamond} = k$-LT since it is the boolean closure of $\mathcal{L}_{LRI_k}$. Henceforth we refer to $\mathcal{L}_{LRI_k^\diamond}$ as the k-Locally Testable (k-LT) languages.

5.6 Generalized subsequence languages

Here we introduce generalized subsequence functions, a general class of functions to which the SP_k and fac_k functions belong. Like those functions, generalized subsequence functions map words to a set of subsequences found within the words. These functions are instantiated by a vector whose number of coordinates determine how many times a subsequence may be discontiguous

and whose coordinate values determine the length of each contiguous part of the subsequence.

Definition 7 For some $n \in \mathbb{N}$, let $\vec{v} = \langle v_0, v_1, \ldots, v_n \rangle$, where each $v_i \in \mathbb{N}$. Let k be the length of the subsequences; i.e. $k = \sum_0^n v_i$.

$$f_{\vec{v}}(w) = \{u \in \Sigma^k : \exists x_0, \ldots, x_n, u_0, \ldots, u_{n+1} \in \Sigma^* \text{ such that } w = u_0 x_0 u_1 x_1, \ldots, u_n x_n u_{n+1} \text{ and } |x_i| = v_i \text{ for all } 0 \leq i \leq n\} \text{ when } k \leq |w|, \text{ and} \{w\} \text{ otherwise}$$

The following examples help make the generalized subsequence functions clear.

Example 6 Let $\vec{v} = \langle 2 \rangle$. Then $f_{\langle 2 \rangle} = fac_2$. Generally, $f_{\langle k \rangle} = fac_k$.

Example 7 Let $\vec{v} = \langle 1, 1 \rangle$. Then $f_{\langle 1,1 \rangle} = SP_2$. Generally, if $\vec{v} = \langle 1, \ldots 1 \rangle$ with $|\vec{v}| = k$. Then $f_{\vec{v}} = SP_k$.

Example 8 Let $\vec{v} = \langle 3, 2, 1 \rangle$ and *a, b, c, d, e, f*$\in \Sigma$. Then $\mathcal{L}_{f_{\langle 3,2,1 \rangle}}$ includes languages which prohibit strings w which contain subsequences $abcdef$ where abc and de must be contiguous in w and $abcdef$ is a subsequence of w.

Generalized subsequence languages make different kinds of distinctions to be made than PT and LT languages. For example, the language in Example 8 is neither k-LT nor k'-PT for any values k, k'. Generalized subsequence languages properly include the k-SP and k-SL classes (Examples 6 and 7), and the boolean closure of the subsequence languages ($f^{\diamond}_{\vec{v}}$) properly includes the LT and PT classes.

Since for any $\vec{v}$, $f_{\vec{v}}$ and $f^{\diamond}_{\vec{v}}$ are string extension functions the learning results in §4 apply. Note that $f_{\vec{v}}(w)$ is computable in time $O(|w|^k)$ where k is the length of the maximal subsequences determined by $\vec{v}$.

6 Other examples

This section provides examples of infinite and nonregular language classes that are string extension learnable. Recall from Theorem 1 that string extension languages are finite unions of blocks of the partition of Σ^* induced by f. Assuming the blocks of this partition can be enumerated, the range of f can be construed as $\mathcal{P}_{fin}(\mathbb{N})$.

grammar G	Language of G
$\emptyset$	$\emptyset$
$\{0\}$	$a^n b^n$
$\{1\}$	$\Sigma^* \backslash a^n b^n$
$\{0,\ 1\}$	Σ^*

Table 3: The language class $\mathcal{L}_f$ from Example 9

In the examples considered so far, the enumeration of the blocks is essentially encoded in particular substrings (or tuples of substrings). However, much less clever enumerations are available.

Example 9 Let A = {0,1} and consider the following function:

$$f(w) = \begin{cases} 0 & \text{iff } w \in a^n b^n \\ 1 & \text{otherwise} \end{cases}$$

The function f belongs to $\mathcal{SEF}$ because it is maps strings to a finite co-domain. $\mathcal{L}_f$ has four languages shown in Table 3.

The language class in Example 9 is not regular because it includes the well-known context-free language $a^n b^n$. This collection of languages is also not closed under reversal.

There are also infinite language classes that are string extension language classes. Arguably the simplest example is the class of finite languages, denoted $\mathcal{L}_{fin}$.

Example 10 Consider the function id which maps words in Σ^* to their singleton sets, i.e. $id(w) = \{w\}$.[5] A grammar G is then a finite subset of Σ^*, and so $L(G)$ is just a finite set of words in Σ^*; in fact, $L(G) = G$. It follows that $\mathcal{L}_{id} = \mathcal{L}_{fin}$.

It can be easily seen that the function id induces the trivial partition over Σ^*, and languages are just finite unions of these blocks. The learner ϕ_{id} makes no generalizations at all, and only remembers what it has observed.

There are other more interesting infinite string extension classes. Here is one relating to the Parikh map (Parikh, 1966). For all $a \in \Sigma$, let $f_a(w)$ be the set containing n where n is the number of times the letter a occurs in the string w. For

[5] Strictly speaking, this is not the identity function per se, but it is as close to the identity function as one can get since string extension functions are defined as mappings from strings to sets. However, once the domain of the function is extended (Equation 1), then it follows that id is the identity function when its argument is a set of strings.

example $f_a(babab) = \{2\}$. Thus f_a is a total function mapping strings to singleton sets of natural numbers, so it is a string extension function. This function induces an infinite partition of Σ^*, where the words in any particular block have the same number of letters a. It is convenient to enumerate the blocks according to how many occurrences of the letter a may occur in words within the block. Hence, B_0 is the block whose words have no occurrences of a, B_1 is the block whose words have one occurrence of a, and so on.

In this case, a grammar G is a finite subset of $\mathbb{N}$, e.g. $\{2, 3, 4\}$. $L(G)$ is simply those words which have either 2, 3, or 4, occurrences of the letter a. Thus $\mathcal{L}_{f_a}$ is an infinite class, which contains languages of infinite size, which is easily identified in the limit from positive data by ϕ_{f_a}.

This section gave examples of nonregular and nonfinite string extension classes by pursuing the implications of Theorem 1, which established that $f \in \mathcal{SEF}$ partition Σ^* into blocks of which languages are finite unions thereof. The string extension function f provides an effective way of encoding all languages L in $\mathcal{L}_f$ because $f(L)$ encodes a finite set, the grammar.

7 Conclusion and open questions

One contribution of this paper is a unified way of thinking about many formal language classes, all of which have been shown to be identifiable in the limit from positive data by a string extension learner. Another contribution is a recipe for defining classes of languages identifiable in the limit from positive data by this kind of learner.

As shown, these learners have many desirable properties. In particular, they are globally consistent, locally conservative, and set-driven. Additionally, the learner is guaranteed to be efficient in the size of the sample, provided the function f itself is efficient in the length of the string.

Several additional questions of interest remain open for theoretical linguistics, theoretical computer science, and computational linguistics.

For theoretical linguistics, it appears that the string extension function $f = (LRI_3, P_2)$, which defines a class of languages which obey restrictions on both contiguous subsequences of length 3 and on discontiguous subsequences of length 2, provides a good first approximation to the segmental phonotactic patterns in natural languages (Heinz, 2007). The string extension learner for this class is essentially two learners: ϕ_{LRI_3} and ϕ_{P_2}, operating simultaneously.[6] The learners make predictions about generalizations, which can be tested in artificial language learning experiments on adults and infants (Rogers and Pullum, to appear; Chambers et al., 2002; Onishi et al., 2003; Cristiá and Seidl, 2008).[7]

For theoretical computer science, it remains an open question what property holds of functions f in $\mathcal{SEF}$ to ensure that $\mathcal{L}_f$ is regular, context-free, or context-sensitive. For known subregular classes, there are constructions that provide deterministic automata that suggest the relevant properties. (See, for example, Garcia et al. (1990) and Garica and Ruiz (1996).)

Also, Timo Kötzing and Samuel Moelius (p.c.) suggest that the results here may be generalized along the following lines. Instead of defining the function f as a map from strings to finite subsets, let f be a function from strings to elements of a lattice. A grammar G is an element of the lattice and the language of the G are all strings w such that f maps w to a grammar less than G. Learners ϕ_f are defined as the least upper bound of its current hypothesis and the grammar to which f maps the current word.[8] Kasprzik and Kötzing (2010) develop this idea and demonstrate additional properties of string extension classes and learning, and show that the pattern languages (Angluin, 1980a) form a string extension class.[9]

Also, hyperplane learning (Clark et al., 2006a; Clark et al., 2006b) and function-distinguishable learning (Fernau, 2003) similarly associate language classes with functions. How those analyses relate to the current one remains open.

Finally, since the stochastic counterpart of k-SL class is the n-gram model, it is plausible that probabilistic string extension language classes can form the basis of new natural language processing techniques. (Heinz and Rogers, 2010) show

[6] This learner resembles what learning theorists call *parallel learning* (Case and Moelius, 2007) and what cognitive scientists call *modular learning* (Gallistel and King, 2009).

[7] I conjecture that morphological and syntactic patterns are generally not amenable to a string extension learning analysis because these patterns appear to require a paradigm, i.e. a set of data points, before any conclusion can be confidently drawn about the generating grammar. Stress patterns also do not appear to be amenable to a string extension learning (Heinz, 2007; Edlefsen et al., 2008; Heinz, 2009).

[8] See also Lange et al. (2008, Theorem 15) and Case et al. (1999, pp.101-103).

[9] The basic idea is to consider the lattice $\mathbb{L} = \langle \mathcal{L}_{fin}, \supseteq \rangle$. Each element of $\mathbb{L}$ is a finite set of strings representing the intersection of all pattern languages consistent with this set.

how to efficiently estimate k-SP distributions, and it is conjectured that the other string extension language classes can be recast as classes of distributions, which can also be successfully estimated from positive evidence.

Acknowledgments

This work was supported by a University of Delaware Research Fund grant during the 2008-2009 academic year. I would like to thank John Case, Alexander Clark, Timo Kötzing, Samuel Moelius, James Rogers, and Edward Stabler for valuable discussion. I would also like to thank Timo Kötzing for careful reading of an earlier draft and for catching some errors. Remaining errors are my responsibility.

References

Dana Angluin. 1980a. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62.

Dana Angluin. 1980b. Inductive inference of formal languages from positive data. *Information Control*, 45:117–135.

Dana Angluin. 1988. Identifying languages from stochastic examples. Technical Report 614, Yale University, New Haven, CT.

D. Beauquier and J.E. Pin. 1991. Languages and scanners. *Theoretical Computer Science*, 84:3–21.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965.

J.A. Brzozowski and I. Simon. 1973. Characterization of locally testable events. *Discrete Math*, 4:243–271.

J.A. Brzozowski. 1962. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical Theory of Automata*, pages 529–561. New York.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Pascal Caron. 2000. Families of locally testable languages. *Theoretical Computer Science*, 242:361–376.

John Case and Sam Moelius. 2007. Parallelism increases iterative learning power. In *18th Annual Conference on Algorithmic Learning Theory (ALT07)*, volume 4754 of *Lecture Notes in Artificial Intelligence*, pages 49–63. Springer-Verlag, Berlin.

John Case, Sanjay Jain, Steffen Lange, and Thomas Zeugmann. 1999. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110.

Kyle E. Chambers, Kristine H. Onishi, and Cynthia Fisher. 2002. Learning phonotactic constraints from brief auditory experience. *Cognition*, 83:B13–B23.

Alexander Clark, Christophe Costa Florêncio, and Chris Watkins. 2006a. Languages as hyperplanes: grammatical inference with string kernels. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 90–101.

Alexander Clark, Christophe Costa Florêncio, Chris Watkins, and Mariette Serayet. 2006b. Planar languages and learnability. In *Proceedings of the 8th International Colloquium on Grammatical Inference (ICGI)*, pages 148–160.

Alejandrina Cristiá and Amanda Seidl. 2008. Phonological features in infants phonotactic learning: Evidence from artificial grammar learning. *Language, Learning, and Development*, 4(3):203–227.

Colin de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138.

Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. 2008. Deciding strictly local (SL) languages. In Jon Breitenbucher, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pages 66–73.

Henning Fernau. 2003. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711.

C.R. Gallistel and Adam Philip King. 2009. *Memory and the Computational Brain*. Wiley-Blackwell.

Pedro García and José Ruiz. 1996. Learning k-piecewise testable languages from positive data. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Interference: Learning Syntax from Sentences*, volume 1147 of *Lecture Notes in Computer Science*, pages 203–210. Springer.

Pedro García and José Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars*, 7:125–140.

Pedro Garcia, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338.

E.M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

J. Grainger and C. Whitney. 2004. Does the huamn mnid raed wrods as a wlohe? *Trends in Cognitive Science*, 8:58–59.

Jeffrey Heinz and James Rogers. 2010. Estimating strictly piecewise distributions. In *Proceedings of the ACL*.

Jeffrey Heinz. 2007. *The Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles.

Jeffrey Heinz. 2009. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351.

Jeffrey Heinz. to appear. Learning long distance phonotactics. *Linguistic Inquiry*.

J. J. Horning. 1969. *A Study of Grammatical Inference*. Ph.D. thesis, Stanford University.

Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. 1999. *Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change)*. The MIT Press, 2nd edition.

Sanjay Jain, Steffen Lange, and Sandra Zilles. 2007. Some natural conditions on incremental learning. *Information and Computation*, 205(11):1671–1684.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition.

Anna Kasprzik and Timo Kötzing. to appear. String extension learning using lattices. In *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, Trier, Germany.

S.M. Kim, R. McNaughton, and R. McCloskey. 1991. A polynomial time algorithm for the local testability problem of deterministic finite automata. *IEEE Trans. Comput.*, 40(10):1087–1093.

Leonid (Aryeh) Kontorovich, Corinna Cortes, and Mehryar Mohri. 2008. Kernel methods for learning languages. *Theoretical Computer Science*, 405(3):223 – 236. Algorithmic Learning Theory.

Steffen Lange, Thomas Zeugmann, and Sandra Zilles. 2008. Learning indexed families of recursive languages from positive data: A survey. *Theoretical Computer Science*, 397:194–232.

H. Lodhi, N. Cristianini, J. Shawe-Taylor, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Language Research*, 2:419–444.

M. Lothaire, editor. 2005. *Applied Combinatorics on Words*. Cmbridge University Press, 2nd edition.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

R. McNaughton. 1974. Algebraic decision procedures for local testability. *Math. Systems Theory*, 8:60–76.

Kristine H. Onishi, Kyle E. Chambers, and Cynthia Fisher. 2003. Infants learn phonotactic regularities from brief auditory experience. *Cognition*, 87:B69–B77.

R. J. Parikh. 1966. On context-free languages. *Journal of the ACM, 13, 570581.*, 13:570–581.

James Rogers and Geoffrey Pullum. to appear. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2009. On languages piecewise testable in the strict sense. In *Proceedings of the 11th Meeting of the Assocation for Mathematics of Language*.

John Shawe-Taylor and Nello Christianini. 2005. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222.

Imre Simon. 1993. The product of rational languages. In *ICALP '93: Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, pages 430–444, London, UK. Springer-Verlag.

Carol Whitney and Piers Cornelissen. 2008. SERIOL reading. *Language and Cognitive Processes*, 23:143–164.

Carol Whitney. 2001. How the brain encodes the order of letters in a printed word: the SERIOL model and selective literature review. *Psychonomic Bulletin Review*, 8:221–243.

Compositional Matrix-Space Models of Language

Sebastian Rudolph
Karlsruhe Institute of Technology
Karlsruhe, Germany
rudolph@kit.edu

Eugenie Giesbrecht
FZI Forschungszentrum Informatik
Karlsuhe, Germany
giesbrecht@fzi.de

Abstract

We propose CMSMs, a novel type of generic compositional models for syntactic and semantic aspects of natural language, based on matrix multiplication. We argue for the structural and cognitive plausibility of this model and show that it is able to cover and combine various common compositional NLP approaches ranging from statistical word space models to symbolic grammar formalisms.

1 Introduction

In computational linguistics and information retrieval, Vector Space Models (Salton et al., 1975) and its variations – such as Word Space Models (Schütze, 1993), Hyperspace Analogue to Language (Lund and Burgess, 1996), or Latent Semantic Analysis (Deerwester et al., 1990) – have become a mainstream paradigm for text representation. Vector Space Models (VSMs) have been empirically justified by results from cognitive science (Gärdenfors, 2000). They embody the distributional hypothesis of meaning (Firth, 1957), according to which the meaning of words is defined by contexts in which they (co-)occur. Depending on the specific model employed, these contexts can be either local (the co-occurring words), or global (a sentence or a paragraph or the whole document). Indeed, VSMs proved to perform well in a number of tasks requiring computation of *semantic relatedness* between words, such as synonymy identification (Landauer and Dumais, 1997), automatic thesaurus construction (Grefenstette, 1994), semantic priming, and word sense disambiguation (Padó and Lapata, 2007).

Until recently, little attention has been paid to the task of modeling more complex conceptual structures with such models, which constitutes a crucial barrier for semantic vector models on the way to model language (Widdows, 2008). An emerging area of research receiving more and more attention among the advocates of distributional models addresses the methods, algorithms, and evaluation strategies for representing *compositional* aspects of language within a VSM framework. This requires novel modeling paradigms, as most VSMs have been predominantly used for meaning representation of single words and the key problem of common bag-of-words-based VSMs is that word order information and thereby the structure of the language is lost.

There are approaches under way to work out a combined framework for meaning representation using both the advantages of symbolic and distributional methods. Clark and Pulman (2007) suggest a conceptual model which unites symbolic and distributional representations by means of traversing the parse tree of a sentence and applying a tensor product for combining vectors of the meanings of words with the vectors of their roles. The model is further elaborated by Clark et al. (2008).

To overcome the aforementioned difficulties with VSMs and work towards a tight integration of symbolic and distributional approaches, we propose a *Compositional Matrix-Space Model* (CMSM) which employs matrices instead of vectors and makes use of matrix multiplication as the one and only composition operation.

The paper is structured as follows: We start by providing the necessary basic notions in linear algebra in Section 2. In Section 3, we give a formal account of the concept of compositionality, introduce our model, and argue for the plausibility of CMSMs in the light of structural and cognitive considerations. Section 4 shows how common VSM approaches to compositionality can be captured by CMSMs while Section 5 illustrates the capabilities of our model to likewise cover symbolic approaches. In Section 6, we demonstrate

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 907–916,
Uppsala, Sweden, 11-16 July 2010.

how several CMSMs can be combined into one model. We provide an overview of related work in Section 7 before we conclude and point out avenues for further research in Section 8.

2 Preliminaries

In this section, we recap some aspects of linear algebra to the extent needed for our considerations about CMSMs. For a more thorough treatise we refer the reader to a linear algebra textbook (such as Strang (1993)).

Vectors. Given a natural number n, an n-dimensional vector $\mathbf{v}$ over the reals can be seen as a list (or tuple) containing n real numbers $r_1, \ldots, r_n \in \mathbb{R}$, written $\mathbf{v} = (r_1 \quad r_2 \quad \cdots \quad r_n)$. Vectors will be denoted by lowercase bold font letters and we will use the notation $\mathbf{v}(i)$ to refer to the ith entry of vector $\mathbf{v}$. As usual, we write $\mathbb{R}^n$ to denote the set of all n-dimensional vectors with real entries. Vectors can be added entry-wise, i.e., $(r_1 \quad \cdots \quad r_n) + (r'_1 \quad \cdots \quad r'_n) = (r_1 + r'_1 \quad \cdots \quad r_n + r'_n)$. Likewise, the entry-wise product (also known as Hadamard product) is defined by $(r_1 \quad \cdots \quad r_n) \odot (r'_1 \quad \cdots \quad r'_n) = (r_1 \cdot r'_1 \quad \cdots \quad r_n \cdot r'_n)$.

Matrices. Given two real numbers n, m, an $n \times m$ matrix over the reals is an array of real numbers with n rows and m columns. We will use capital letters to denote matrices and, given a matrix M we will write $M(i, j)$ to refer to the entry in the ith row and the jth column:

$$M = \begin{pmatrix} M(1,1) & M(1,2) & \cdots & M(1,j) & \cdots & M(1,m) \\ M(2,1) & M(2,2) & & & & \vdots \\ \vdots & & & & & \vdots \\ M(i,1) & & & M(i,j) & & \vdots \\ \vdots & & & & & \vdots \\ M(n,1) & M(1,2) & \cdots & \cdots & \cdots & M(n,m) \end{pmatrix}$$

The set of all $n \times m$ matrices with real number entries is denoted by $\mathbb{R}^{n \times m}$. Obviously, m-dimensional vectors can be seen as $1 \times m$ matrices. A matrix can be *transposed* by exchanging columns and rows: given the $n \times m$ matrix M, its transposed version M^T is a $m \times n$ matrix defined by $M^T(i, j) = M(j, i)$.

Linear Mappings. Beyond being merely array-like data structures, matrices correspond to certain type of functions, so-called *linear mappings*, having vectors as in- and output. More precisely, an $n \times m$ matrix M applied to an m-dimensional vector $\mathbf{v}$ yields an n-dimensional vector $\mathbf{v}'$ (written: $\mathbf{v}M = \mathbf{v}'$) according to

$$\mathbf{v}'(i) = \sum_{j=1}^{m} \mathbf{v}(j) \cdot M(i, j)$$

Linear mappings can be concatenated, giving rise to the notion of standard matrix multiplication: we write $M_1 M_2$ to denote the matrix that corresponds to the linear mapping defined by applying first M_1 and then M_2. Formally, the matrix product of the $n \times l$ matrix M_1 and the $l \times m$ matrix M_2 is an $n \times m$ matrix $M = M_1 M_2$ defined by

$$M(i, j) = \sum_{k=1}^{l} M_1(i, k) \cdot M_2(k, j)$$

Note that the matrix product is associative (i.e., $(M_1 M_2)M_3 = M_1(M_2 M_3)$ always holds, thus parentheses can be omitted) but not commutative ($M_1 M_2 = M_2 M_1$ does not hold in general, i.e., the order matters).

Permutations. Given a natural number n, a *permutation* on $\{1 \ldots n\}$ is a bijection (i.e., a mapping that is one-to-one and onto) $\Phi : \{1 \ldots n\} \to \{1 \ldots n\}$. A permutation can be seen as a "reordering scheme" on a list with n elements: the element at position i will get the new position $\Phi(i)$ in the reordered list. Likewise, a permutation can be applied to a vector resulting in a rearrangement of the entries. We write Φ^n to denote the permutation corresponding to the n-fold application of Φ and Φ^{-1} to denote the permutation that "undoes" Φ.

Given a permutation Φ, the corresponding *permutation matrix* M_Φ is defined by

$$M_\Phi(i, j) = \begin{cases} 1 \text{ if } \Phi(j) = i, \\ 0 \text{ otherwise.} \end{cases}$$

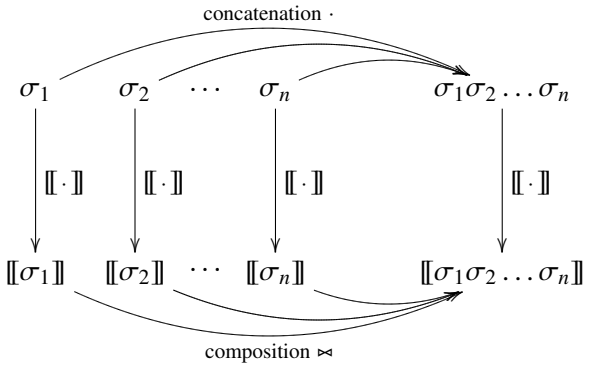
Then, obviously permuting a vector according to Φ can be expressed in terms of matrix multiplication as well as we obtain for any vector $\mathbf{v} \in \mathbb{R}^n$:

$$\Phi(\mathbf{v}) = \mathbf{v} M_\Phi$$

Likewise, iterated application (Φ^n) and the inverses Φ^{-n} carry over naturally to the corresponding notions in matrices.

3 Compositionality and Matrices

The underlying principle of compositional semantics is that the meaning of a sentence (or a word phrase) can be derived from the meaning of its constituent tokens by applying a composition operation. More formally, the underlying idea can be described as follows: given a mapping $[\![\cdot]\!] : \Sigma \to \mathbb{S}$ from a set of tokens (words) Σ into some semantical space $\mathbb{S}$ (the elements of which we will simply call "meanings"), we find a semantic composition operation $\bowtie$: $\mathbb{S}^* \to \mathbb{S}$ mapping sequences of meanings to meanings such that the meaning of a sequence of tokens $\sigma_1\sigma_2 \ldots \sigma_n$ can be obtained by applying $\bowtie$ to the sequence $[\![\sigma_1]\!][\![\sigma_2]\!] \ldots [\![\sigma_n]\!]$. This situation qualifies $[\![\cdot]\!]$ as a homomorphism between $(\Sigma^*, \cdot)$ and $(\mathbb{S}, \bowtie)$ and can be displayed as follows:



A great variety of linguistic models are subsumed by this general idea ranging from purely symbolic approaches (like type systems and categorial grammars) to rather statistical models (like vector space and word space models). At the first glance, the underlying encodings of word semantics as well as the composition operations differ significantly. However, we argue that a great variety of them can be incorporated – and even freely inter-combined – into a unified model where the semantics of simple tokens and complex phrases is expressed by matrices and the composition operation is standard matrix multiplication.

More precisely, in Compositional Marix-Space Models, we have $\mathbb{S} = \mathbb{R}^{n \times n}$, i.e. the semantical space consists of quadratic matrices, and the composition operator $\bowtie$ coincides with matrix multiplication as introduced in Section 2. In the following, we will provide diverse arguments illustrating that CMSMs are intuitive and natural.

3.1 Algebraic Plausibility – Structural Operation Properties

Most linear-algebra-based operations that have been proposed to model composition in language models are associative and commutative. Thereby, they realize a multiset (or bag-of-words) semantics that makes them insensitive to structural differences of phrases conveyed through word order.

While associativity seems somewhat acceptable and could be defended by pointing to the stream-like, sequential nature of language, commutativity seems way less justifiable, arguably.

As mentioned before, matrix multiplication is associative but non-commutative, whence we propose it as more adequate for modeling compositional semantics of language.

3.2 Neurological Plausibility – Progression of Mental States

From a very abstract and simplified perspective, CMSMs can also be justified neurologically.

Suppose the mental state of a person at one specific moment in time can be encoded by a vector $\mathbf{v}$ of numerical values; one might, e.g., think of the level of excitation of neurons. Then, an external stimulus or signal, such as a perceived word, will result in a change of the mental state. Thus, the external stimulus can be seen as a function being applied to $\mathbf{v}$ yielding as result the vector $\mathbf{v}'$ that corresponds to the persons mental state after receiving the signal. Therefore, it seems sensible to associate with every signal (in our case: token σ) a respective function (a linear mapping, represented by a matrix $M = [\![\sigma]\!]$ that maps mental states to mental states (i.e. vectors $\mathbf{v}$ to vectors $\mathbf{v}' = \mathbf{v}M$).

Consequently, the subsequent reception of inputs σ, σ' associated to matrices M and M' will transform a mental vector $\mathbf{v}$ into the vector $(\mathbf{v}M)M'$ which by associativity equals $\mathbf{v}(MM')$. Therefore, MM' represents the mental state transition triggered by the signal sequence $\sigma\sigma'$. Naturally, this consideration carries over to sequences of arbitrary length. This way, abstracting from specific initial mental state vectors, our semantic space $\mathbb{S}$ can be seen as a function space of mental transformations represented by matrices, whereby matrix multiplication realizes subsequent execution of those transformations triggered by the input token sequence.

3.3 Psychological Plausibility – Operations on Working Memory

A structurally very similar argument can be provided on another cognitive explanatory level. There have been extensive studies about human language processing justifying the hypothesis of a *working memory* (Baddeley, 2003). The mental state vector can be seen as representation of a person's working memory which gets transformed by external input. Note that matrices can perform standard memory operations such as storing, deleting, copying etc. For instance, the matrix $M_{\text{copy(k,l)}}$ defined by

$$M_{\text{copy(k,l)}}(i,j) = \begin{cases} 1 \text{ if } i = j \neq l \text{ or } i = k,\ j = l, \\ 0 \text{ otherwise.} \end{cases}$$

applied to a vector $\mathbf{v}$, will copy its kth entry to the lth position. This mechanism of storage and insertion can, e.g., be used to simulate simple forms of anaphora resolution.

4 CMSMs Encode Vector Space Models

In VSMs numerous vector operations have been used to model composition (Widdows, 2008), some of the more advanced ones being related to quantum mechanics. We show how these common composition operators can be modeled by CMSMs.[1] Given a vector composition operation $\bowtie\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, we provide a surjective function $\psi_\bowtie : \mathbb{R}^n \to \mathbb{R}^{n' \times n'}$ that translates the vector representation into a matrix representation in a way such that for all $\mathbf{v}_1, \ldots \mathbf{v}_k \in \mathbb{R}^n$ holds

$$\mathbf{v}_1 \bowtie \ldots \bowtie \mathbf{v}_k = \psi_\bowtie^{-1}(\psi_\bowtie(\mathbf{v}_1) \ldots \psi_\bowtie(\mathbf{v}_k))$$

where $\psi_\bowtie(\mathbf{v}_i)\psi_\bowtie(\mathbf{v}_j)$ denotes matrix multiplication of the matrices assigned to $\mathbf{v}_i$ and $\mathbf{v}_j$.

4.1 Vector Addition

As a simple basic model for semantic composition, vector addition has been proposed. Thereby, tokens σ get assigned (usually high-dimensional) vectors $\mathbf{v}_\sigma$ and to obtain a representation of the meaning of a phrase or a sentence $w = \sigma_1 \ldots \sigma_k$, the vector sum of the vectors associated to the constituent tokens is calculated: $\mathbf{v}_w = \sum_{i=1}^k \mathbf{v}_{\sigma_i}$.

[1] In our investigations we will focus on VSM composition operations which preserve the format (i.e. which yield a vector of the same dimensionality), as our notion of compositionality requires models that allow for iterated composition. In particular, this rules out dot product and tensor product. However the convolution product can be seen as a condensed version of the tensor product.

This kind of composition operation is subsumed by CMSMs; suppose in the original model, a token σ gets assigned the vector $\mathbf{v}_\sigma$, then by defining

$$\psi_+(\mathbf{v}_\sigma) = \left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline & \mathbf{v}_\sigma & & 1 \end{array}\right)$$

(mapping n-dimensional vectors to $(n+1)\times(n+1)$ matrices), we obtain for a phrase $w = \sigma_1 \ldots \sigma_k$

$$\psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} + \ldots + \mathbf{v}_{\sigma_k} = \mathbf{v}_w.$$

Proof. By induction on k. For $k = 1$, we have $\mathbf{v}_w = \mathbf{v}_\sigma = \psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}))$. For $k > 1$, we have

$$\begin{aligned} & \psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_{k-1}})\psi_+(\mathbf{v}_{\sigma_k})) \\ = \ & \psi_+^{-1}(\psi_+(\psi_+^{-1}(\psi_+(\mathbf{v}_{\sigma_1}) \ldots \psi_+(\mathbf{v}_{\sigma_{k-1}})))\psi_+(\mathbf{v}_{\sigma_k})) \\ \overset{i.h.}{=} \ & \psi_+^{-1}(\psi_+(\textstyle\sum_{i=1}^{k-1} \mathbf{v}_{\sigma_i})\psi_+(\mathbf{v}_{\sigma_k})) \end{aligned}$$

$$= \psi_+^{-1}\left(\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \sum_{i=1}^{k-1}\mathbf{v}_{\sigma_i}(1) & \cdots & \sum_{i=1}^{k-1}\mathbf{v}_{\sigma_i}(n) & 1 \end{array}\right)\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \mathbf{v}_{\sigma_k}(1) & \cdots & \mathbf{v}_{\sigma_k}(n) & 1 \end{array}\right)\right)$$

$$= \psi_+^{-1}\left(\begin{array}{ccc|c} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & & 1 & 0 \\ \hline \sum_{i=1}^{k}\mathbf{v}_{\sigma_i}(1) & \cdots & \sum_{i=1}^{k}\mathbf{v}_{\sigma_i}(n) & 1 \end{array}\right) = \sum_{i=1}^{k}\mathbf{v}_{\sigma_i} \qquad q.e.d.[2]$$

4.2 Component-wise Multiplication

On the other hand, the Hadamard product (also called entry-wise product, denoted by $\odot$) has been proposed as an alternative way of semantically composing token vectors.

By using a different encoding into matrices, CMSMs can simulate this type of composition operation as well. By letting

$$\psi_\odot(\mathbf{v}_\sigma) = \begin{pmatrix} \mathbf{v}_\sigma(1) & 0 & \cdots & 0 \\ 0 & \mathbf{v}_\sigma(2) & & \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{v}_\sigma(n) \end{pmatrix},$$

we obtain an $n \times n$ matrix representation for which $\psi_\odot^{-1}(\psi_\odot(\mathbf{v}_{\sigma_1}) \ldots \psi_\odot(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} \odot \ldots \odot \mathbf{v}_{\sigma_k} = \mathbf{v}_w$.

4.3 Holographic Reduced Representations

Holographic reduced representations as introduced by Plate (1995) can be seen as a refinement

[2] The proofs for the respective correspondences for $\odot$ and $\circledast$ as well as the permutation-based approach in the following sections are structurally analog, hence, we will omit them for space reasons.

of convolution products with the benefit of preserving dimensionality: given two vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n$, their *circular convolution product* $\mathbf{v}_1 \circledast \mathbf{v}_2$ is again an n-dimensional vector $\mathbf{v}_3$ defined by

$$\mathbf{v}_3(i+1) = \sum_{k=0}^{n-1} \mathbf{v}_1(k+1) \cdot \mathbf{v}_2((i-k \mod n)+1)$$

for $0 \leq i \leq n-1$. Now let $\psi_{\circledast}(\mathbf{v})$ be the $n \times n$ matrix M with

$$M(i,j) = \mathbf{v}((j-i \mod n)+1).$$

In the 3-dimensional case, this would result in

$$\psi_{\circledast}(\mathbf{v}(1) \quad \mathbf{v}(2) \quad \mathbf{v}(3)) = \begin{pmatrix} \mathbf{v}(1) & \mathbf{v}(2) & \mathbf{v}(3) \\ \mathbf{v}(3) & \mathbf{v}(1) & \mathbf{v}(2) \\ \mathbf{v}(2) & \mathbf{v}(3) & \mathbf{v}(1) \end{pmatrix}$$

Then, it can be readily checked that

$$\psi_{\circledast}^{-1}(\psi_{\circledast}(\mathbf{v}_{\sigma_1}) \ldots \psi_{\circledast}(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{\sigma_1} \circledast \ldots \circledast \mathbf{v}_{\sigma_k} = \mathbf{v}_w.$$

4.4 Permutation-based Approaches

Sahlgren et al. (2008) use permutations on vectors to account for word order. In this approach, given a token σ_m occurring in a sentence $w = \sigma_1 \ldots \sigma_k$ with predefined "uncontextualized" vectors $\mathbf{v}_{\sigma_1} \ldots \mathbf{v}_{\sigma_k}$, we compute the contextualized vector $\mathbf{v}_{w,m}$ for σ_m by

$$\mathbf{v}_{w,m} = \Phi^{1-m}(\mathbf{v}_{\sigma_1}) + \ldots + \Phi^{k-m}(\mathbf{v}_{\sigma_k}),$$

which can be equivalently transformed into

$$\Phi^{1-m}(\mathbf{v}_{\sigma_1} + \Phi(\ldots + \Phi(\mathbf{v}_{\sigma_{k-1}} + (\Phi(\mathbf{v}_{\sigma_k}))) \ldots)).$$

Note that the approach is still token-centered, i.e., a vector representation of a token is endowed with contextual representations of surrounding tokens. Nevertheless, this setting can be transferred to a CMSM setting by recording the position of the focused token as an additional parameter. Now, by assigning every $\mathbf{v}_\sigma$ the matrix

$$\psi_\Phi(\mathbf{v}_\sigma) = \left(\begin{array}{c|c} M_\Phi & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \mathbf{v}_\sigma & 1 \end{array}\right)$$

we observe that for

$$M_{w,m} := (M_\Phi^-)^{m-1} \psi_\Phi(\mathbf{v}_{\sigma_1}) \ldots \psi_\Phi(\mathbf{v}_{\sigma_k})$$

we have

$$M_{w,m} = \left(\begin{array}{c|c} M_\Phi^{k-m} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \mathbf{v}_{w,m} & 1 \end{array}\right),$$

whence $\psi_\Phi^{-1}((M_\Phi^-)^{m-1} \psi_\Phi(\mathbf{v}_{\sigma_1}) \ldots \psi_\Phi(\mathbf{v}_{\sigma_k})) = \mathbf{v}_{w,m}$.

5 CMSMs Encode Symbolic Approaches

Now we will elaborate on symbolic approaches to language, i.e., discrete grammar formalisms, and show how they can conveniently be embedded into CMSMs. This might come as a surprise, as the apparent likeness of CMSMs to vector-space models may suggest incompatibility to discrete settings.

5.1 Group Theory

Group theory and grammar formalisms based on groups and pre-groups play an important role in computational linguistics (Dymetman, 1998; Lambek, 1958). From the perspective of our compositionality framework, those approaches employ a group (or pre-group) $(G, \cdot)$ as semantical space $\mathbb{S}$ where the group operation (often written as multiplication) is used as composition operation $\bowtie$.

According Cayley's Theorem (Cayley, 1854), every group G is isomorphic to a permutation group on some set S. Hence, assuming finiteness of G and consequently S, we can encode group-based grammar formalisms into CMSMs in a straightforward way by using permutation matrices of size $|S| \times |S|$.

5.2 Regular Languages

Regular languages constitute a basic type of languages characterized by a symbolic formalism. We will show how to select the assignment $[\![\cdot]\!]$ for a CMSM such that the matrix associated to a token sequence exhibits whether this sequence belongs to a given regular language, that is if it is accepted by a given finite state automaton. As usual (cf. e.g., Hopcroft and Ullman (1979)) we define a nondeterministic finite automaton $\mathcal{A} = (Q, \Sigma, \Delta, Q_\mathrm{I}, Q_\mathrm{F})$ with $Q = \{q_0, \ldots, q_{n-1}\}$ being the set of states, Σ the input alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ the transition relation, and Q_I and Q_F being the sets of initial and final states, respectively.

Then we assign to every token $\sigma \in \Sigma$ the $n \times n$ matrix $[\![\sigma]\!] = M$ with

$$M(i,j) = \begin{cases} 1 \text{ if } (q_i, \sigma, q_j) \in \Delta, \\ 0 \text{ otherwise.} \end{cases}$$

Hence essentially, the matrix M encodes all state transitions which can be caused by the input σ. Likewise, for a word $w = \sigma_1 \dots \sigma_k \in \Sigma^*$, the matrix $M_w := [\![\sigma_1]\!] \dots [\![\sigma_k]\!]$ will encode all state transitions mediated by w. Finally, if we define vectors $\mathbf{v}_\mathrm{I}$ and $\mathbf{v}_\mathrm{F}$ by

$$\mathbf{v}_\mathrm{I}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{I}, \\ 0 \text{ otherwise,} \end{cases} \qquad \mathbf{v}_\mathrm{F}(i) = \begin{cases} 1 \text{ if } q_i \in Q_\mathrm{F}, \\ 0 \text{ otherwise,} \end{cases}$$

then we find that w is accepted by $\mathcal{A}$ exactly if $\mathbf{v}_\mathrm{I} M_w \mathbf{v}_\mathrm{F}^T \geq 1$.

5.3 The General Case: Matrix Grammars

Motivated by the above findings, we now define a general notion of matrix grammars as follows:

Definition 1 *Let Σ be an alphabet. A* matrix grammar *$\mathcal{M}$ of degree n is defined as the pair $\langle [\![\cdot]\!], AC\rangle$ where $[\![\cdot]\!]$ is a mapping from Σ to $n \times n$ matrices and $AC = \{\langle \mathbf{v}'_1, \mathbf{v}_1, r_1\rangle, \dots, \langle \mathbf{v}'_m, \mathbf{v}_m, r_m\rangle\}$ with $\mathbf{v}'_1, \mathbf{v}_1, \dots, \mathbf{v}'_m, \mathbf{v}_m \in \mathbb{R}^n$ and $r_1, \dots, r_m \in \mathbb{R}$ is a finite set of* acceptance conditions. *The language generated by $\mathcal{M}$ (denoted by $L(\mathcal{M})$) contains a token sequence $\sigma_1 \dots \sigma_k \in \Sigma^*$ exactly if $\mathbf{v}'_i [\![\sigma_1]\!] \dots [\![\sigma_k]\!] \mathbf{v}_i^T \geq r_i$ for all $i \in \{1, \dots, m\}$. We will call a language L* matricible *if $L = L(\mathcal{M})$ for some matrix grammar $\mathcal{M}$.*

Then, the following proposition is a direct consequence from the preceding section.

Proposition 1 *Regular languages are matricible.*

However, as demonstrated by the subsequent examples, also many non-regular and even non-context-free languages are matricible, hinting at the expressivity of our grammar model.

Example 1 *We define $\mathcal{M}\langle [\![\cdot]\!], AC\rangle$ with*

$$\Sigma = \{a, b, c\} \qquad [\![a]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$[\![b]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad [\![c]\!] = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 3 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} AC = \{ &\langle (0 \quad 0 \quad 1 \quad 1), (1 \quad -1 \quad 0 \quad 0), 0\rangle, \\ &\langle (0 \quad 0 \quad 1 \quad 1), (-1 \quad 1 \quad 0 \quad 0), 0\rangle\} \end{aligned}$$

Then $L(\mathcal{M})$ contains exactly all palindromes from $\{a, b, c\}^*$, i.e., the words $d_1 d_2 \dots d_{n-1} d_n$ for which $d_1 d_2 \dots d_{n-1} d_n = d_n d_{n-1} \dots d_2 d_1$.

Example 2 *We define $\mathcal{M} = \langle [\![\cdot]\!], AC\rangle$ with*

$$\Sigma = \{a, b, c\} \qquad [\![a]\!] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$[\![b]\!] = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad [\![c]\!] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\begin{aligned} AC = \{ &\langle (1 \ 0 \ 0 \ 0 \ 0 \ 0), (0 \ 0 \ 1 \ 0 \ 0 \ 0), 1\rangle, \\ &\langle (0 \ 0 \ 0 \ 1 \ 1 \ 0), (0 \ 0 \ 0 \ 1 \ -1 \ 0), 0\rangle, \\ &\langle (0 \ 0 \ 0 \ 0 \ 1 \ 1), (0 \ 0 \ 0 \ 0 \ 1 \ -1), 0\rangle, \\ &\langle (0 \ 0 \ 0 \ 1 \ 1 \ 0), (0 \ 0 \ 0 \ -1 \ 0 \ 1), 0\rangle\} \end{aligned}$$

Then $L(\mathcal{M})$ is the (non-context-free) language $\{a^m b^m c^m \mid m > 0\}$.

The following properties of matrix grammars and matricible language are straightforward.

Proposition 2 *All languages characterized by a set of linear equations on the letter counts are matricible.*

Proof. Suppose $\Sigma = \{a_1, \dots a_n\}$. Given a word w, let x_i denote the number of occurrences of a_i in w. A linear equation on the letter counts has the form

$$k_1 x_1 + \dots + k_n x_n = k \qquad (k, k_1, \dots, k_n \in \mathbb{R})$$

Now define $[\![a_i]\!] = \psi_+(\mathbf{e}_i)$, where $\mathbf{e}_i$ is the ith unit vector, i.e. it contains a 1 at he ith position and 0 in all other positions. Then, it is easy to see that w will be mapped to $M = \psi_+(x_1 \quad \cdots \quad x_n)$. Due to the fact that $\mathbf{e}_{n+1} M = (x_1 \quad \cdots \quad x_n \quad 1)$ we can enforce the above linear equation by defining the acceptance conditions

$$\begin{aligned} AC = \{ &\langle \mathbf{e}_{n+1}, (k_1 \quad \dots \quad k_n \quad -k), 0\rangle, \\ &\langle -\mathbf{e}_{n+1}, (k_1 \quad \dots \quad k_n \quad -k), 0\rangle\}. \end{aligned}$$

q.e.d.

Proposition 3 *The intersection of two matricible languages is again a matricible language.*

Proof. This is a direct consequence of the considerations in Section 6 together with the observation, that the new set of acceptance conditions is trivially obtained from the old ones with adapted dimensionalities. *q.e.d.*

Note that the fact that the language $\{a^m b^m c^m \mid m > 0\}$ is matricible, as demonstrated in Example 2 is a straightforward consequence of the Propositions 1, 2, and 3, since the language in question can be described as the intersection of the regular language $a^+b^+c^+$ with the language characterized by the equations $x_a - x_b = 0$ and $x_b - x_c = 0$. We proceed by giving another account of the expressivity of matrix grammars by showing undecidability of the emptiness problem.

Proposition 4 *The problem whether there is a word which is accepted by a given matrix grammar is undecidable.*

Proof. The undecidable *Post correspondence problem* (Post, 1946) is described as follows: given two lists of words $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ over some alphabet Σ', is there a sequence of numbers $h_1, \ldots, h_m$ $(1 \leq h_j \leq n)$ such that $u_{h_1} \ldots u_{h_m} = v_{h_1} \ldots v_{h_m}$?

We now reduce this problem to the emptiness problem of a matrix grammar. W.l.o.g., let $\Sigma' = \{a_1, \ldots, a_k\}$. We define a bijection # from Σ'^* to $\mathbb{N}$ by

$$\#(a_{n_1} a_{n_2} \ldots a_{n_l}) = \sum_{i=1}^{l} (n_i - 1) \cdot k^{(l-i)}$$

Note that this is indeed a bijection and that for $w_1, w_2 \in \Sigma'^*$, we have

$$\#(w_1 w_2) = \#(w_1) \cdot k^{|w_2|} + \#(w_2).$$

Now, we define $\mathcal{M}$ as follows:

$$\Sigma = \{b_1, \ldots b_n\} \qquad [\![b_i]\!] = \begin{pmatrix} k^{|u_i|} & 0 & 0 \\ 0 & k^{|v_i|} & 0 \\ \#(u_i) & \#(v_i) & 1 \end{pmatrix}$$

$$AC = \{ \langle (0 \quad 0 \quad 1), (1 \quad -1 \quad 0), 0 \rangle, \\ \langle (0 \quad 0 \quad 1), (-1 \quad 1 \quad 0), 0 \rangle \}$$

Using the above fact about # and a simple induction on m, we find that

$$[\![a_{h_1}]\!] \ldots [\![a_{h_m}]\!] = \begin{pmatrix} k^{|u_{h_1} .. u_{h_m}|} & 0 & 0 \\ 0 & k^{|v_{h_1} .. v_{h_m}|} & 0 \\ \#(u_{h_1} \ldots u_{h_m}) & \#(v_{h_1} \ldots v_{h_m}) & 1 \end{pmatrix}$$

Evaluating the two acceptance conditions, we find them satisfied exactly if $\#(u_{h_1} \ldots u_{h_m}) = \#(v_{h_1} \ldots v_{h_m})$. Since # is a bijection, this is the case if and only if $u_{h_1} \ldots u_{h_m} = v_{h_1} \ldots v_{h_m}$. Therefore $\mathcal{M}$ accepts $b_{h_1} \ldots b_{h_m}$ exactly if the sequence $h_1, \ldots, h_m$ is a solution to the given Post Correspondence Problem. Consequently, the question whether such a solution exists is equivalent to the question whether the language $L(\mathcal{M})$ is non-empty. *q.e.d.*

These results demonstrate that matrix grammars cover a wide range of formal languages. Nevertheless some important questions remain open and need to be clarified next:

Are all context-free languages matricible? We conjecture that this is not the case.[3] Note that this question is directly related to the question whether Lambek calculus can be modeled by matrix grammars.

Are matricible languages closed under concatenation? That is: given two arbitrary matricible languages L_1, L_2, is the language $L = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ again matricible? Being a property common to all language types from the Chomsky hierarchy, answering this question is surprisingly non-trivial for matrix grammars.

In case of a negative answer to one of the above questions it might be worthwhile to introduce an extended notion of context grammars to accommodate those desirable properties. For example, allowing for some nondeterminism by associating several matrices to one token would ensure closure under concatenation.

How do the theoretical properties of matrix grammars depend on the underlying algebraic structure? Remember that we considered matrices containing real numbers as entries. In general, matrices can be defined on top of any mathematical structure that is (at least) a semiring (Golan, 1992). Examples for semirings are the natural numbers, boolean algebras, or polynomials with natural number coefficients. Therefore, it would be interesting to investigate the influence of the choice of the underlying semiring on the properties of the matrix grammars – possibly non-standard structures turn out to be more appropriate for capturing certain compositional language properties.

6 Combination of Different Approaches

Another central advantage of the proposed matrix-based models for word meaning is that several matrix models can be easily combined into one.

[3] For instance, we have not been able to find a matrix grammar that recognizes the language of all well-formed parenthesis expressions.

Again assume a sequence $w = \sigma_1 \dots \sigma_k$ of tokens with associated matrices $[\![\sigma_1]\!], \dots, [\![\sigma_k]\!]$ according to one specific model and matrices $(\!|\sigma_1|\!), \dots, (\!|\sigma_k|\!)$ according to another.

Then we can combine the two models into one $\{\!|\ \cdot\ |\!\}$ by assigning to σ_i the matrix

$$\{\!|\sigma_i|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_i]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_i|\!) & \\ 0 & & 0 & & & \end{array}\right)$$

By doing so, we obtain the correspondence

$$\{\!|\sigma_1|\!\} \dots \{\!|\sigma_k|\!\} = \left(\begin{array}{ccc|ccc} & & & 0 & \cdots & 0 \\ & [\![\sigma_1]\!] \dots [\![\sigma_k]\!] & & \vdots & \ddots & \\ & & & 0 & & 0 \\ \hline 0 & \cdots & 0 & & & \\ \vdots & \ddots & & & (\!|\sigma_1|\!) \dots (\!|\sigma_k|\!) & \\ 0 & & 0 & & & \end{array}\right)$$

In other words, the semantic compositions belonging to two CMSMs can be executed "in parallel." Mark that by providing non-zero entries for the upper right and lower left matrix part, information exchange between the two models can be easily realized.

7 Related Work

We are not the first to suggest an extension of classical VSMs to matrices. Distributional models based on matrices or even higher-dimensional arrays have been proposed in information retrieval (Gao et al., 2004; Antonellis and Gallopoulos, 2006). However, to the best of our knowledge, the approach of realizing compositionality via matrix multiplication seems to be entirely original.

Among the early attempts to provide more compelling combinatory functions to capture word order information and the non-commutativity of linguistic compositional operation in VSMs is the work of Kintsch (2001) who is using a more sophisticated addition function to model predicate-argument structures in VSMs.

Mitchell and Lapata (2008) formulate semantic composition as a function $m = f(w_1, w_2, R, K)$ where R is a relation between w_1 and w_2 and K is additional knowledge. They evaluate the model with a number of addition and multiplication operations for vector combination on a sentence similarity task proposed by Kintsch (2001). Widdows (2008) proposes a number of more advanced vector operations well-known from quantum mechanics, such as tensor product and convolution, to model composition in vector spaces. He shows the ability of VSMs to reflect the relational and phrasal meanings on a simplified analogy task. Giesbrecht (2009) evaluates four vector composition operations (+, $\odot$, tensor product, convolution) on the task of identifying multi-word units. The evaluation results of the three studies are not conclusive in terms of which vector operation performs best; the different outcomes might be attributed to the underlying word space models; e.g., the models of Widdows (2008) and Giesbrecht (2009) feature dimensionality reduction while that of Mitchell and Lapata (2008) does not. In the light of these findings, our CMSMs provide the benefit of just one composition operation that is able to mimic all the others as well as combinations thereof.

8 Conclusion and Future Work

We have introduced a generic model for compositionality in language where matrices are associated with tokens and the matrix representation of a token sequence is obtained by iterated matrix multiplication. We have given algebraic, neurological, and psychological plausibility indications in favor of this choice. We have shown that the proposed model is expressive enough to cover and combine a variety of distributional and symbolic aspects of natural language. This nourishes the hope that matrix models can serve as a kind of *lingua franca* for compositional models.

This having said, some crucial questions remain before CMSMs can be applied in practice:
How to acquire CMSMs for large token sets and specific purposes? We have shown the value and expressivity of CMSMs by providing carefully hand-crafted encodings. In practical cases, however, the number of token-to-matrix assignments will be too large for this manual approach. Therefore, methods to (semi-)automatically acquire those assignments from available data are required. To this end, machine learning techniques need to be investigated with respect to their applicability to this task. Presumably, hybrid approaches have to be considered, where parts of

the matrix representation are learned whereas others are stipulated in advance guided by external sources (such as lexical information).

In this setting, data sparsity may be overcome through tensor methods: given a set T of tokens together with the matrix assignment $[\![]\!] : T \rightarrow \mathbb{R}^{n \times n}$, this datastructure can be conceived as a 3-dimensional array (also known as tensor) of size $n \times n \times |T|$ wherein the single token-matrices can be found as slices. Then tensor decomposition techniques can be applied in order to find a compact representation, reduce noise, and cluster together similar tokens (Tucker, 1966; Rendle et al., 2009). First evaluation results employing this approach to the task of free associations are reported by Giesbrecht (2010).

How does linearity limit the applicability of CMSMs? In Section 3, we justified our model by taking the perspective of tokens being functions which realize mental state transitions. Yet, using matrices to represent those functions restricts them to linear mappings. Although this restriction brings about benefits in terms of computability and theoretical accessibility, the limitations introduced by this assumption need to be investigated. Clearly, certain linguistic effects (like a-posteriori disambiguation) cannot be modeled via linear mappings. Instead, we might need some in-between application of simple nonlinear functions in the spirit of quantum-collapsing of a "superposed" mental state (such as the winner takes it all, survival of the top-k vector entries, and so forth). Thus, another avenue of further research is to generalize from the linear approach.

Acknowledgements

This work was supported by the German Research Foundation (DFG) under the Multipla project (grant 38457858) as well as by the German Federal Ministry of Economics (BMWi) under the project Theseus (number 01MQ07019).

References

[Antonellis and Gallopoulos2006] Ioannis Antonellis and Efstratios Gallopoulos. 2006. Exploring term-document matrices from matrix models in text mining. *CoRR*, abs/cs/0602076.

[Baddeley2003] Alan D. Baddeley. 2003. Working memory and language: An overview. *Journal of Communication Disorder*, 36:198–208.

[Cayley1854] Arthur Cayley. 1854. On the theory of groups as depending on the symbolic equation $\theta^n = 1$. *Philos. Magazine*, 7:40–47.

[Clark and Pulman2007] Stephen Clark and Stephen Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction, Stanford, CA, 2007*, pages 52–55.

[Clark et al.2008] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pages 133–140.

[Deerwester et al.1990] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[Dymetman1998] Marc Dymetman. 1998. Group theory and computational linguistics. *J. of Logic, Lang. and Inf.*, 7(4):461–497.

[Firth1957] John R. Firth. 1957. A synopsis of linguistic theory 1930-55. *Studies in linguistic analysis*, pages 1–32.

[Gao et al.2004] Kai Gao, Yongcheng Wang, and Zhiqi Wang. 2004. An efficient relevant evaluation model in information retrieval and its application. In *CIT '04: Proceedings of the The Fourth International Conference on Computer and Information Technology*, pages 845–850. IEEE Computer Society.

[Gärdenfors2000] Peter Gärdenfors. 2000. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA, USA.

[Giesbrecht2009] Eugenie Giesbrecht. 2009. In search of semantic compositionality in vector spaces. In Sebastian Rudolph, Frithjof Dau, and Sergei O. Kuznetsov, editors, *ICCS*, volume 5662 of *Lecture Notes in Computer Science*, pages 173–184. Springer.

[Giesbrecht2010] Eugenie Giesbrecht. 2010. Towards a matrix-based distributional model of meaning. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Student Research Workshop*. ACL.

[Golan1992] Jonathan S. Golan. 1992. *The theory of semirings with applications in mathematics and theoretical computer science*. Addison-Wesley Longman Ltd.

[Grefenstette1994] Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Springer.

[Hopcroft and Ullman1979] John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[Kintsch2001] Walter Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

[Lambek1958] Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

[Landauer and Dumais1997] Thomas K. Landauer and Susan T. Dumais. 1997. Solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, (104).

[Lund and Burgess1996] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208.

[Mitchell and Lapata2008] Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. ACL.

[Padó and Lapata2007] Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

[Plate1995] Tony Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

[Post1946] Emil L. Post. 1946. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268.

[Rendle et al.2009] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *KDD*, pages 727–736. ACM.

[Sahlgren et al.2008] Magnus Sahlgren, Anders Holst, and Pentti Kanerva. 2008. Permutations as a means to encode order in word space. In *Proc. CogSci'08*, pages 1300–1305.

[Salton et al.1975] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Schütze1993] Hinrich Schütze. 1993. Word space. In Lee C. Giles, Stephen J. Hanson, and Jack D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan-Kaufmann.

[Strang1993] Gilbert Strang. 1993. *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

[Tucker1966] Ledyard R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3).

[Widdows2008] Dominic Widdows. 2008. Semantic vector products: some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

Cross-Language Document Summarization Based on Machine Translation Quality Prediction

Xiaojun Wan, Huiying Li and Jianguo Xiao
Institute of Compute Science and Technology, Peking University, Beijing 100871, China
Key Laboratory of Computational Linguistics (Peking University), MOE, China
{wanxiaojun,lihuiying,xiaojianguo}@icst.pku.edu.cn

Abstract

Cross-language document summarization is a task of producing a summary in one language for a document set in a different language. Existing methods simply use machine translation for document translation or summary translation. However, current machine translation services are far from satisfactory, which results in that the quality of the cross-language summary is usually very poor, both in readability and content. In this paper, we propose to consider the translation quality of each sentence in the English-to-Chinese cross-language summarization process. First, the translation quality of each English sentence in the document set is predicted with the SVM regression method, and then the quality score of each sentence is incorporated into the summarization process. Finally, the English sentences with high translation quality and high informativeness are selected and translated to form the Chinese summary. Experimental results demonstrate the effectiveness and usefulness of the proposed approach.

1 Introduction

Given a document or document set in one source language, cross-language document summarization aims to produce a summary in a different target language. In this study, we focus on English-to-Chinese document summarization for the purpose of helping Chinese readers to quickly understand the major content of an English document or document set. This task is very important in the field of multilingual information access.

Till now, most previous work focuses on monolingual document summarization, but cross-language document summarization has received little attention in the past years. A straightforward way for cross-language document summarization is to translate the summary from the source language to the target language by using machine translation services. However, though machine translation techniques have been advanced a lot, the machine translation quality is far from satisfactory, and in many cases, the translated texts are hard to understand. Therefore, the translated summary is likely to be hard to understand by readers, i.e., the summary quality is likely to be very poor. For example, the translated Chinese sentence for an ordinary English sentence ("It is also Mr Baker who is making the most of presidential powers to dispense largesse.") by using Google Translate is "同时，也是贝克是谁提出了对总统权力免除最慷慨。". The translated sentence is hard to understand because it contains incorrect translations and it is very disfluent. If such sentences are selected into the summary, the quality of the summary would be very poor.

In order to address the above problem, we propose to consider the translation quality of the English sentences in the summarization process. In particular, the translation quality of each English sentence is predicted by using the SVM regression method, and then the predicted MT quality score of each sentence is incorporated into the sentence evaluation process, and finally both informative and easy-to-translate sentences are selected and translated to form the Chinese summary.

An empirical evaluation is conducted to evaluate the performance of machine translation quality prediction, and a user study is performed to evaluate the cross-language summary quality. The results demonstrate the effectiveness of the proposed approach.

The rest of this paper is organized as follows: Section 2 introduces related work. The system is overviewed in Section 3. In Sections 4 and 5, we present the detailed algorithms and evaluation

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 917–926,
Uppsala, Sweden, 11-16 July 2010.

results of machine translation quality prediction and cross-language summarization, respectively. We discuss in Section 6 and conclude this paper in Section 7.

2 Related Work

2.1 Machine Translation Quality Prediction

Machine translation evaluation aims to assess the correctness and quality of the translation. Usually, the human reference translation is provided, and various methods and metrics have been developed for comparing the system-translated text and the human reference text. For example, the BLEU metric, the NIST metric and their relatives are all based on the idea that the more shared substrings the system-translated text has with the human reference translation, the better the translation is. Blatz et al. (2003) investigate training sentence-level confidence measures using a variety of fuzzy match scores. Albrecht and Hwa (2007) rely on regression algorithms and reference-based features to measure the quality of sentences.

Transition evaluation without using reference translations has also been investigated. Quirk (2004) presents a supervised method for training a sentence level confidence measure on translation output using a human-annotated corpus. Features derived from the source sentence and the target sentence (e.g. sentence length, perplexity, etc.) and features about the translation process are leveraged. Gamon et al. (2005) investigate the possibility of evaluating MT quality and fluency at the sentence level in the absence of reference translations, and they can improve on the correlation between language model perplexity scores and human judgment by combing these perplexity scores with class probabilities from a machine-learned classifier. Specia et al. (2009) use the ICM theory to identify the threshold to map a continuous predicted score into "good" or "bad" categories. Chae and Nenkova (2009) use surface syntactic features to assess the fluency of machine translation results.

In this study, we further predict the translation quality of an English sentence before the machine translation process, i.e., we do not leverage reference translation and the target sentence.

2.2 Document Summarization

Document summarization methods can be generally categorized into extraction-based methods and abstraction-based methods. In this paper, we focus on extraction-based methods. Extraction-based summarization methods usually assign each sentence a saliency score and then rank the sentences in a document or document set.

For single document summarization, the sentence score is usually computed by empirical combination of a number of statistical and linguistic feature values, such as term frequency, sentence position, cue words, stigma words, topic signature (Luhn 1969; Lin and Hovy, 2000). The summary sentences can also be selected by using machine learning methods (Kupiec et al., 1995; Amini and Gallinari, 2002) or graph-based methods (ErKan and Radev, 2004; Mihalcea and Tarau, 2004). Other methods include mutual reinforcement principle (Zha 2002; Wan et al., 2007).

For multi-document summarization, the centroid-based method (Radev et al., 2004) is a typical method, and it scores sentences based on cluster centroids, position and TFIDF features. NeATS (Lin and Hovy, 2002) makes use of new features such as topic signature to select important sentences. Machine Learning based approaches have also been proposed for combining various sentence features (Wong et al., 2008). The influences of input difficulty on summarization performance have been investigated in (Nenkova and Louis, 2008). Graph-based methods have also been used to rank sentences in a document set. For example, Mihalcea and Tarau (2005) extend the TextRank algorithm to compute sentence importance in a document set. Cluster-level information has been incorporated in the graph model to better evaluate sentences (Wan and Yang, 2008). Topic-focused or query biased multi-document summarization has also been investigated (Wan et al., 2006). Wan et al. (2010) propose the EUSUM system for extracting easy-to-understand English summaries for non-native readers.

Several pilot studies have been performed for the cross-language summarization task by simply using document translation or summary translation. Leuski et al. (2003) use machine translation for English headline generation for Hindi documents. Lim et al. (2004) propose to generate a Japanese summary without using a Japanese summarization system, by first translating Japanese documents into Korean documents, and then extracting summary sentences by using Korean summarizer, and finally mapping Korean summary sentences to Japanese summary sentences. Chalendar et al. (2005) focuses on semantic analysis and sentence generation techniques for cross-language summarization. Orasan

and Chiorean (2008) propose to produce summaries with the MMR method from Romanian news articles and then automatically translate the summaries into English. Cross language query based summarization has been investigated in (Pingali et al., 2007), where the query and the documents are in different languages. Other related work includes multilingual summarization (Lin et al., 2005), which aims to create summaries from multiple sources in multiple languages. Siddharthan and McKeown (2005) use the information redundancy in multilingual input to correct errors in machine translation and thus improve the quality of multilingual summaries.

3 The Proposed Approach

Previous methods for cross-language summarization usually consist of two steps: one step for summarization and one step for translation. Different order of the two steps can lead to the following two basic English-to-Chinese summarization methods:

Late Translation (LateTrans): Firstly, an English summary is produced for the English document set by using existing summarization methods. Then, the English summary is automatically translated into the corresponding Chinese summary by using machine translation services.

Early Translation (EarlyTrans): Firstly, the English documents are translated into Chinese documents by using machine translation services. Then, a Chinese summary is produced for the translated Chinese documents.

Generally speaking, the LateTrans method has a few advantages over the EarlyTrans method:

1) The LateTrans method is much more efficient than the EarlyTrans method, because only a very few summary sentences are required to be translated in the LateTrans method, whereas all the sentences in the documents are required to be translated in the EarlyTrans method.

2) The LateTrans method is deemed to be more effective than the EarlyTrans method, because the translation errors of the sentences have great influences on the summary sentence extraction in the EarlyTrans method.

Thus in this study, we adopt the LateTrans method as our baseline method. We also adopt the late translation strategy for our proposed approach.

In the baseline method, a translated Chinese sentence is selected into the summary because the original English sentence is informative. However, an informative and fluent English sentence is likely to be translated into an uninformative and disfluent Chinese sentence, and therefore, this sentence cannot be selected into the summary.

In order to address the above problem of existing methods, our proposed approach takes into account a novel factor of each sentence for cross-language summary extraction. Each English sentence is associated with a score indicating its translation quality. An English sentence with high translation quality score is more likely to be selected into the original English summary, and such English summary can be translated into a better Chinese summary. Figure 1 gives the architecture of our proposed approach.

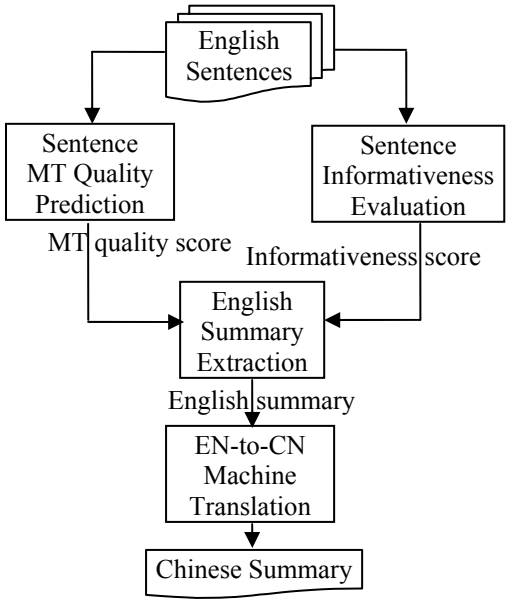


Figure 1: Architecture of our proposed approach

Seen from the figure, our proposed approach consists of four main steps: 1) The machine translation quality score of each English sentence is predicted by using regression methods; 2) The informativeness score of each English sentence is computed by using existing methods; 3) The English summary is produced by making use of both the machine translation quality score and the informativeness score; 4) The extracted English summary is translated into Chinese summary by using machine translation services.

In this study, we adopt *Google Translate*[1] for English-to-Chinese translation. *Google Translate* is one of the state-of-the-art commercial machine translation systems used today. It applies statistical learning techniques to build a translation

[1] http://translate.google.com/translate_t

model based on both monolingual text in the target language and aligned text consisting of examples of human translations between the languages.

The first step and the evaluation results will be described in Section 4, and the other steps and the evaluation results will be described together in Section 5.

4 Machine Translation Quality Prediction

4.1 Methodology

In this study, machine translation (MT) quality reflects both the translation accuracy and the fluency of the translated sentence. An English sentence with high MT quality score is likely to be translated into an accurate and fluent Chinese sentence, which can be easily read and understand by Chinese readers. The MT quality prediction is a task of mapping an English sentence to a numerical value corresponding to a quality level. The larger the value is, the more accurately and fluently the sentence can be translated into Chinese sentence.

As introduced in Section 2.1, several related work has used regression and classification methods for MT quality prediction without reference translations. In our approach, the MT quality of each sentence in the documents is also predicted without reference translations. The difference between our task and previous work is that previous work can make use of both features in source sentence and features in target sentence, while our task only leverages features in source sentence, because in the late translation strategy, the English sentences in the documents have not been translated yet at this step.

In this study, we adopt the ε-support vector regression (ε-SVR) method (Vapnik 1995) for the sentence-level MT quality prediction task. The SVR algorithm is firmly grounded in the framework of statistical learning theory (VC theory). The goal of a regression algorithm is to fit a flat function to the given training data points.

Formally, given a set of training data points $D=\{(x_i,y_i)|\ i=1,2,\ldots,n\} \subset R^d \times R$, where x_i is input feature vector and y_i is associated score, the goal is to fit a function f which approximates the relation inherited between the data set points. The standard form is:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2} w^T w + C\sum_{i=1}^{n} \xi_i + C\sum_{i=1}^{n} \xi_i^*$$

Subject to

$$w^T f(x_i) + b - y_i \le \varepsilon + \xi_i$$
$$y_i - w^T f(x_i) - b \le \varepsilon + \xi_i^*$$
$$\varepsilon, \xi_i, \xi_i^* \ge 0,\ i = 1,\ldots,n.$$

The constant $C>0$ is a parameter for determining the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

In the experiments, we use the LIBSVM tool (Chang and Lin, 2001) with the RBF kernel for the task, and we use the parameter selection tool of 10-fold cross validation via grid search to find the best parameters on the training set with respect to mean squared error (MSE), and then use the best parameters to train on the whole training set.

We use the following two groups of features for each sentence: the first group includes several basic features, and the second group includes several parse based features[2]. They are all derived based on the source English sentence.

The basic features are as follows:

1) **Sentence length**: It refers to the number of words in the sentence.
2) **Sub-sentence number**: It refers to the number of sub-sentences in the sentence. We simply use the punctuation marks as indicators of sub-sentences.
3) **Average sub-sentence length**: It refers to the average number of words in the sub-sentences within the sentence.
4) **Percentage of nouns and adjectives**: It refers to the percentage of noun words or adjective words in the in the sentence.
5) **Number of question words**: It refers to the number of question words (who, whom, whose, when, where, which, how, why, what) in the sentence.

We use the Stanford Lexicalized Parser (Klein and Manning, 2002) with the provided English PCFG model to parse a sentence into a parse tree. The output tree is a context-free phrase structure grammar representation of the sentence. The parse features are then selected as follows:

1) **Depth of the parse tree**: It refers to the depth of the generated parse tree.
2) **Number of SBARs in the parse tree**: SBAR is defined as a clause introduced by a (possibly empty) subordinating conjunction. It is an indictor of sentence complexity.

[2] Other features, including n-gram frequency, perplexity features, etc., are not useful in our study. MT features are not used because *Google Translate* is used as a black box.

3) **Number of NPs in the parse tree**: It refers to the number of noun phrases in the parse tree.
4) **Number of VPs in the parse tree**: It refers to the number of verb phrases in the parse tree.

All the above feature values are scaled by using the provided svm-scale program.

At this step, each English sentence s_i can be associated with a MT quality score *TransScore*(s_i) predicted by the ε-SVR method. The score is finally normalized by dividing by the maximum score.

4.2 Evaluation

4.2.1 Evaluation Setup

In the experiments, we first constructed the gold-standard dataset in the following way:

DUC2001 provided 309 English news articles for document summarization tasks, and the articles were grouped into 30 document sets. The news articles were selected from TREC-9. We chose five document sets (d04, d05, d06, d08, d11) with 54 news articles out of the DUC2001 document sets. The documents were then split into sentences and we used 1736 sentences for evaluation. All the sentences were automatically translated into Chinese sentences by using the *Google Translate* service.

Two Chinese college students were employed for data annotation. They read the original English sentence and the translated Chinese sentence, and then manually labeled the overall translation quality score for each sentence, separately. The translation quality is an overall measure for both the translation accuracy and the readability of the translated sentence. The score ranges between 1 and 5, and 1 means "very bad", and 5 means "very good", and 3 means "normal". The correlation between the two sets of labeled scores is 0.646. The final translation quality score was the average of the scores provided by the two annotators.

After annotation, we randomly separated the labeled sentence set into a training set of 1428 sentences and a test set of 308 sentences. We then used the LIBSVM tool for training and testing.

Two metrics were used for evaluating the prediction results. The two metrics are as follows:

Mean Square Error (MSE): This metric is a measure of how correct each of the prediction values is on average, penalizing more severe errors more heavily. Given the set of prediction scores for the test sentences: $\hat{Y} = \{\hat{y}_i \mid i = 1,...n\}$, and the manually assigned scores for the sentences: $Y = \{y_i \mid i = 1,...n\}$, the MSE of the prediction result is defined as

$$MSE(\hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Pearson's Correlation Coefficient (ρ): This metric is a measure of whether the trends of prediction values matched the trends for human-labeled data. The coefficient between Y and $\hat{Y}$ is defined as

$$\rho = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{n s_y s_{\hat{y}}}$$

where $\bar{y}$ and $\bar{\hat{y}}$ are the sample means of Y and $\hat{Y}$, s_y and $s_{\hat{y}}$ are the sample standard deviations of Y and $\hat{Y}$.

4.2.2 Evaluation Results

Table 1 shows the prediction results. We can see that the overall results are promising. And the correlation is moderately high. The results are acceptable because we only make use of the features derived from the source sentence. The results guarantee that the use of MT quality scores in the summarization process is feasible.

We can also see that both the basic features and the parse features are beneficial to the overall prediction results.

Feature Set	MSE	ρ
Basic features	0.709	0.399
Parse features	0.702	0.395
All features	**0.683**	**0.433**

Table 1: Prediction results

5 Cross-Language Document Summarization

5.1 Methodology

In this section, we first compute the informativeness score for each sentence. The score reflect how the sentence expresses the major topic in the documents. Various existing methods can be used for computing the score. In this study, we adopt the centroid-based method.

The centroid-based method is the algorithm used in the MEAD system. The method uses a heuristic and simple way to sum the sentence scores computed based on different features. The score for each sentence is a linear combination of

the weights computed based on the following three features:

Centroid-based Weight. The sentences close to the centroid of the document set are usually more important than the sentences farther away. And the centroid weight $C(s_i)$ of a sentence s_i is calculated as the cosine similarity between the sentence text and the concatenated text for the whole document set D. The weight is then normalized by dividing the maximal weight.

Sentence Position. The leading several sentences of a document are usually important. So we calculate for each sentence a weight to reflect its position priority as $P(s_i)=1-(i-1)/n$, where i is the sequence of the sentence s_i and n is the total number of sentences in the document. Obviously, i ranges from 1 to n.

First Sentence Similarity. Because the first sentence of a document is very important, a sentence similar to the first sentence is also important. Thus we use the cosine similarity value between a sentence and the corresponding first sentence in the same document as the weight $F(s_i)$ for sentence s_i.

After all the above weights are calculated for each sentence, we sum all the weights and get the overall score for the sentence as follows:

$$InfoScore(s_i) = \alpha \cdot C(s_i) + \beta \cdot P(s_i) + \gamma \cdot F(s_i)$$

where α, β and γ are parameters reflecting the importance of different features. We empirically set $\alpha=\beta=\gamma=1$.

After the informativeness scores for all sentences are computed, the score of each sentence is normalized by dividing by the maximum score.

After we obtain the MT quality score and the informativeness score of each sentence in the document set, we linearly combine the two scores to get the overall score of each sentence.

Formally, let $TransScore(s_i) \in [0,1]$ and $InfoScore(s_i) \in [0,1]$ denote the MT quality score and the informativeness score of sentence s_i, the overall score of the sentence is:

$$OverallScore(s_i) = (1-\lambda) \times InfoScore(s_i) + \lambda \times TransScore(s_i)$$

where $\lambda \in [0,1]$ is a parameter controlling the influences of the two factors. If λ is set to 0, the summary is extracted without considering the MT quality factor. In the experiments, we empirically set the parameter to 0.3 in order to balance the two factors of content informativeness and translation quality.

For multi-document summarization, some sentences are highly overlapping with each other, and thus we apply the same greedy algorithm in (Wan et al., 2006) to penalize the sentences highly overlapping with other highly scored sentences, and finally the informative, novel, and easy-to-translate sentences are chosen into the English summary.

Finally, the sentences in the English summary are translated into the corresponding Chinese sentences by using *Google Translate*, and the Chinese summary is formed.

5.2 Evaluation

5.2.1 Evaluation Setup

In this experiment, we used the document sets provided by DUC2001 for evaluation. As mentioned in Section 4.2.1, DUC2001 provided 30 English document sets for generic multi-document summarization. The average document number per document set was 10. The sentences in each article have been separated and the sentence information has been stored into files. Generic reference English summaries were provided by NIST annotators for evaluation. In our study, we aimed to produce Chinese summaries for the English document sets. The summary length was limited to five sentences, i.e. each summary consisted of five sentences.

The DUC2001 dataset was divided into the following two datasets:

Ideal Dataset: We have manually labeled the MT quality scores for the sentences in five document sets (d04-d11), and we directly used the manually labeled scores in the summarization process. The ideal dataset contained these five document sets.

Real Dataset: The MT quality scores for the sentences in the remaining 25 document sets were automatically predicted by using the learned SVM regression model. And we used the automatically predicted scores in the summarization process. The real dataset contained these 25 document sets.

We performed two evaluation procedures: one based on the ideal dataset to validate the feasibility of the proposed approach, and the other based on the real dataset to demonstrate the effectiveness of the proposed approach in real applications.

To date, various methods and metrics have been developed for English summary evaluation by comparing system summary with reference summary, such as the pyramid method (Nenkova et al., 2007) and the ROUGE metrics (Lin and Hovy, 2003). However, such methods or metrics cannot be directly used for evaluating Chinese summary without reference Chinese summary.

Instead, we developed an evaluation protocol as follows:

The evaluation was based on human scoring. Four Chinese college students participated in the evaluation as subjects. We have developed a friendly tool for helping the subjects to evaluate each Chinese summary from the following three aspects:

Content: This aspect indicates how much a summary reflects the major content of the document set. After reading a summary, each user can select a score between 1 and 5 for the summary. 1 means "very uninformative" and 5 means "very informative".

Readability: This aspect indicates the readability level of the whole summary. After reading a summary, each user can select a score between 1 and 5 for the summary. 1 means "hard to read", and 5 means "easy to read".

Overall: This aspect indicates the overall quality of a summary. After reading a summary, each user can select a score between 1 and 5 for the summary. 1 means "very bad", and 5 means "very good".

We performed the evaluation procedures on the ideal dataset and the read dataset, separately. During each evaluation procedure, we compared our proposed approach (λ=0.3) with the baseline approach without considering the MT quality factor (λ=0). And the two summaries produced by the two systems for the same document set were presented in the same interface, and then the four subjects assigned scores to each summary after they read and compared the two summaries. And the assigned scores were finally averaged across the documents sets and across the subjects.

5.2.2 Evaluation Results

Table 2 shows the evaluation results on the ideal dataset with 5 document sets. We can see that based on the manually labeled MT quality scores, the Chinese summaries produced by our proposed approach are significantly better than that produced by the baseline approach over all three aspects. All subjects agree that our proposed approach can produce more informative and easy-to-read Chinese summaries than the baseline approach.

Table 3 shows the evaluation results on the real dataset with 25 document sets. We can see that based on the automatically predicted MT quality scores, the Chinese summaries produced by our proposed approach are significantly better than that produced by the baseline approach over the readability aspect and the overall aspect. Almost all subjects agree that our proposed approach can produce more easy-to-read and high-quality Chinese summaries than the baseline approach.

Comparing the evaluation results in the two tables, we can find that the performance difference between the two approaches on the ideal dataset is bigger than that on the real dataset, especially on the content aspect. The results demonstrate that the more accurate the MT quality scores are, the more significant the performance improvement is.

Overall, the proposed approach is effective to produce good-quality Chinese summaries for English document sets.

	Baseline Approach			**Proposed Approach**		
	content	**readability**	**overall**	**content**	**readability**	**overall**
Subject1	3.2	2.6	2.8	3.4	3.0	3.4
Subject2	3.0	3.2	3.2	3.4	3.6	3.4
Subject3	3.4	2.8	3.2	3.6	3.8	3.8
Subject4	3.2	3.0	3.2	3.8	3.8	3.8
Average	3.2	2.9	3.1	3.55*	3.55*	3.6*

Table 2: Evaluation results on the ideal dataset (5 document sets)

	Baseline Approach			**Proposed Approach**		
	content	**readability**	**overall**	**content**	**readability**	**overall**
Subject1	2.64	2.56	2.60	2.80	3.24	2.96
Subject2	3.60	2.76	3.36	3.52	3.28	3.64
Subject3	3.52	3.72	3.44	3.56	3.80	3.48
Subject4	3.16	2.96	3.12	3.16	3.44	3.52
Average	3.23	3.00	3.13	3.26	3.44*	3.40*

Table 3: Evaluation results on the real dataset (25 document sets)

(* indicates the difference between the average score of the proposed approach and that of the baseline approach is statistically significant by using t-test.)

5.2.3 *Example Analysis*

In this section, we give two running examples to better show the effectiveness of our proposed approach. The Chinese sentences and the original English sentences in the summary are presented together. The normalized MT quality score for each sentence is also given at the end of the Chinese sentence.

Document set 1: D04 from the ideal dataset

Summary by baseline approach:

s1: 预计美国的保险公司支付，估计在佛罗里达州的73亿美元（37亿英镑），作为安德鲁飓风的结果-迄今为止最昂贵的灾难曾经面临产业。(0.56)
(US INSURERS expect to pay out an estimated Dollars 7.3bn (Pounds 3.7bn) in Florida as a result of Hurricane Andrew - by far the costliest disaster the industry has ever faced.)
s2: 有越来越多的迹象表明安德鲁飓风，不受欢迎的，因为它的佛罗里达和路易斯安那州的受灾居民，最后可能不伤害到连任的布什总统竞选。(0.67)
(THERE are growing signs that Hurricane Andrew, unwelcome as it was for the devastated inhabitants of Florida and Louisiana, may in the end do no harm to the re-election campaign of President George Bush.)
s3: 一般事故发生后，英国著名保险公司昨日表示，保险索赔的安德鲁飓风所引发的成本也高达4000万美元'。 (0.44)
(GENERAL ACCIDENT said yesterday that insurance claims arising from Hurricane Andrew could 'cost it as much as Dollars 40m'.)
s4: 在巴哈马，政府发言人麦库里说，4人死亡已离岛东部群岛报告。 (0.56)
(In the Bahamas, government spokesman Mr Jimmy Curry said four deaths had been reported on outlying eastern islands.)
s5: 新奥尔良的和1.6万人，是特别脆弱，因为该市位于海平面以下，有密西西比河通过其中心的运行和一个大型湖泊立即向北方。(0.44)
(New Orleans, with a population of 1.6m, is particularly vulnerable because the city lies below sea level, has the Mississippi River running through its centre and a large lake immediately to the north.)

Summary by proposed approach:

s1: 预计美国的保险公司支付，估计在佛罗里达州的73亿美元（37亿英镑），作为安德鲁飓风的结果-迄今为止最昂贵的灾难曾经面临产业。(0.56)
(US INSURERS expect to pay out an estimated Dollars 7.3bn (Pounds 3.7bn) in Florida as a result of Hurricane Andrew - by far the costliest disaster the industry has ever faced.)
s2: 有越来越多的迹象表明安德鲁飓风，不受欢迎的，因为它的佛罗里达和路易斯安那州的受灾居民，最后可能不伤害到连任的布什总统竞选。(0.67)
(THERE are growing signs that Hurricane Andrew, unwelcome as it was for the devastated inhabitants of Florida and Louisiana, may in the end do no harm to the re-election campaign of President George Bush.)
s3: 在巴哈马，政府发言人麦库里说，4人死亡已离岛东部群岛报告。(0.56)
(In the Bahamas, government spokesman Mr Jimmy Curry said four deaths had been reported on outlying eastern islands.)
s4: 在首当其冲的损失可能会集中在美国的保险公司，业内分析人士昨天说。 (0.89)
(The brunt of the losses are likely to be concentrated among US insurers, industry analysts said yesterday.)
s5: 在北迈阿密，损害是最小的。(1.0)
(In north Miami, damage is minimal.)

Document set 2: D54 from the real dataset

Summary by baseline approach:

s1: 两个加州11月6日投票的主张，除其他限制外，全州成员及州议员的条件。(0.57)
(Two propositions on California's Nov. 6 ballot would, among other things, limit the terms of statewide officeholders and state legislators.)
s2: 原因之一是任期限制将开放到现在的政治职务任职排除了许多人的职业生涯。(0.36)
(One reason is that term limits would open up politics to many people now excluded from office by career incumbents.)
s3: 建议限制国会议员及州议员都很受欢迎，越来越多的条件是，根据专家和投票。(0.20)
(Proposals to limit the terms of members of Congress and of state legislators are popular and getting more so, according to the pundits and the polls.)
s4: 国家法规的酒吧首先从运行时间为国会候选人已举行了加入的资格规定了宪法规定，并已失效。(0.24)
(State statutes that bar first-time candidates from running for Congress have been held to add to the qualifications set forth in the Constitution and have been invalidated.)
s5: 另一个论点是，公民的同时，不断进入新的华盛顿国会将面临流动更好的结果，比政府的任期较长的代表提供的。(0.20)
(Another argument is that a citizen Congress with its continuing flow of fresh faces into Washington would result in better government than that provided by representatives with lengthy tenure.)

Summary by proposed approach:

s1: 两个加州 11 月 6 日投票的主张，除其他限制外，全州成员及州议员的条件。(0.57)
(Two propositions on California's Nov. 6 ballot would, among other things, limit the terms of statewide officeholders and state legislators.)
s2: 原因之一是任期限制将开放到现在的政治职务任职排除了许多人的职业生涯。(0.36)
(One reason is that term limits would open up politics to many people now excluded from office by career incumbents.)
s3: 另一个论点是，公民的同时，不断进入新的华盛顿国会将面临流动更好的结果，比政府的任期较长的代表提供的。(0.20)
(Another argument is that a citizen Congress with its continuing flow of fresh faces into Washington would result in better government than that provided by representatives with lengthy tenure.)
s4: 有两个国会任期限制，经济学家，至少公共选择那些劝说，要充分理解充分的理由。(0.39)
(There are two solid reasons for congressional term limitation that economists, at least those of the public-choice persuasion, should fully appreciate.)
s5: 与国会的问题的根源是，除非有重大丑闻，几乎是不可能战胜现任。(0.47)
(The root of the problems with Congress is that, barring major scandal, it is almost impossible to defeat an incumbent.)

6 Discussion

In this study, we adopt the late translation strategy for cross-document summarization. As mentioned earlier, the late translation strategy has some advantages over the early translation strategy. However, in the early translation strategy, we can use the features derived from both the source English sentence and the target Chinese sentence to improve the MT quality prediction results.

Overall, the framework of our proposed approach can be easily adapted for cross-document summarization with the early translation strategy.

And an empirical comparison between the two strategies is left as our future work.

Though this study focuses on English-to-Chinese document summarization, cross-language summarization tasks for other languages can also be solved by using our proposed approach.

7 Conclusion and Future Work

In this study we propose a novel approach to address the cross-language document summarization task. Our proposed approach predicts the MT quality score of each English sentence and then incorporates the score into the summarization process. The user study results verify the effectiveness of the approach.

In future work, we will manually translate English reference summaries into Chinese reference summaries, and then adopt the ROUGE metrics to perform automatic evaluation of the extracted Chinese summaries by comparing them with the Chinese reference summaries. Moreover, we will further improve the sentence's MT quality by using sentence compression or sentence reduction techniques.

Acknowledgments

This work was supported by NSFC (60873155), Beijing Nova Program (2008B03), NCET (NCET-08-0006), RFDP (20070001059) and National High-tech R&D Program (2008AA01Z421). We thank the students for participating in the user study. We also thank the anonymous reviewers for their useful comments.

References

J. Albrecht and R. Hwa. 2007. A re-examination of machine learning approaches for sentence-level mt evaluation. In *Proceedings of ACL2007*.

M. R. Amini, P. Gallinari. 2002. The Use of Unlabeled Data to Improve Supervised Learning for Text Summarization. In *Proceedings of SIGIR2002*.

J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. 2003. Confidence estimation for statistical machine translation. *Johns Hopkins Summer Workshop Final Report*.

J. Chae and A. Nenkova. 2009. Predicting the fluency of text with shallow structural features: case studies of machine translation and human-written text. In *Proceedings of EACL2009*.

G. de Chalendar, R. Besançon, O. Ferret, G. Grefenstette, and O. Mesnard. 2005. Crosslingual summarization with thematic extraction, syntactic sentence simplification, and bilingual generation. In *Workshop on Crossing Barriers in Text Summarization Research, 5th International Conference on Recent Advances in Natural Language Processing (RANLP2005)*.

C.-C. Chang and C.-J. Lin. 2001. LIBSVM : a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

G. ErKan, D. R. Radev. LexPageRank. 2004. Prestige in Multi-Document Text Summarization. In *Proceedings of EMNLP2004*.

M. Gamon, A. Aue, and M. Smets. 2005. Sentence-level MT evaluation without reference translations: beyond language modeling. In *Proceedings of EAMT2005*.

D. Klein and C. D. Manning. 2002. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Proceedings of NIPS2002*.

J. Kupiec, J. Pedersen, F. Chen. 1995. A.Trainable Document Summarizer. In *Proceedings of SIGIR1995*.

A. Leuski, C.-Y. Lin, L. Zhou, U. Germann, F. J. Och, E. Hovy. 2003. Cross-lingual C*ST*RD: English access to Hindi information. *ACM Transactions on Asian Language Information Processing*, 2(3): 245-269.

J.-M. Lim, I.-S. Kang, J.-H. Lee. 2004. Multi-document summarization using cross-language texts. In *Proceedings of NTCIR-4*.

C. Y. Lin, E. Hovy. 2000. The Automated Acquisition of Topic Signatures for Text Summarization. In *Proceedings of the 17th Conference on Computational Linguistics*.

C..-Y. Lin and E.. H. Hovy. 2002. From Single to Multi-document Summarization: A Prototype System and its Evaluation. In *Proceedings of ACL-02*.

C.-Y. Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of HLT-NAACL -03*.

C.-Y. Lin, L. Zhou, and E. Hovy. 2005. Multilingual summarization evaluation 2005: automatic evaluation report. In *Proceedings of MSE (ACL-2005 Workshop)*.

H. P. Luhn. 1969. The Automatic Creation of literature Abstracts. *IBM Journal of Research and Development*, 2(2).

R. Mihalcea, P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP2004*.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP-05*.

A. Nenkova and A. Louis. 2008. Can you summarize this? Identifying correlates of input difficulty for generic multi-document summarization. In *Proceedings of ACL-08:HLT*.

A. Nenkova, R. Passonneau, and K. McKeown. 2007. The Pyramid method: incorporating human content selection variation in summarization evaluation.

ACM Transactions on Speech and Language Processing (TSLP), 4(2).

C. Orasan, and O. A. Chiorean. 2008. Evaluation of a Crosslingual Romanian-English Multi-document Summariser. In *Proceedings of 6th Language Resources and Evaluation Conference (LREC2008)*.

P. Pingali, J. Jagarlamudi and V. Varma. 2007. Experiments in cross language query focused multi-document summarization. In *Workshop on Cross Lingual Information Access Addressing the Information Need of Multilingual Societies in IJCAI2007*.

C. Quirk. 2004. Training a sentence-level machine translation confidence measure. In *Proceedings of LREC2004*.

D. R. Radev, H. Y. Jing, M. Stys and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919-938.

A. Siddharthan and K. McKeown. 2005. Improving multilingual summarization: using redundancy in the input to correct MT errors. In *Proceedings of HLT/EMNLP-2005*.

L. Specia, Z. Wang, M. Turchi, J. Shawe-Taylor, C. Saunders. 2009. Improving the Confidence of Machine Translation Quality Estimates. In *MT Summit 2009 (Machine Translation Summit XII)*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

X. Wan, H. Li and J. Xiao. 2010. EUSUM: extracting easy-to-understand English summaries for non-native readers. In *Proceedings of SIGIR2010*.

X. Wan, J. Yang and J. Xiao. 2006. Using cross-document random walks for topic-focused multi-documetn summarization. In *Proceedings of WI2006*.

X. Wan and J. Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR-08*.

X. Wan, J. Yang and J. Xiao. 2007. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. In *Proceedings of ACL2007*.

K.-F. Wong, M. Wu and W. Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of COLING-08*.

H. Y. Zha. 2002. Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. In *Proceedings of SIGIR2002*.

A new Approach to Improving Multilingual Summarization using a Genetic Algorithm

Marina Litvak
Ben-Gurion University of the Negev
Beer Sheva, Israel
litvakm@bgu.ac.il

Mark Last
Ben-Gurion University of the Negev
Beer Sheva, Israel
mlast@bgu.ac.il

Menahem Friedman
Ben-Gurion University of the Negev
Beer Sheva, Israel
fmenahem@bgu.ac.il

Abstract

Automated summarization methods can be defined as "language-independent," if they are not based on any language-specific knowledge. Such methods can be used for multilingual summarization defined by Mani (2001) as "processing several languages, with summary in the same language as input." In this paper, we introduce MUSE, a language-independent approach for extractive summarization based on the linear optimization of several sentence ranking measures using a genetic algorithm. We tested our methodology on two languages—English and Hebrew—and evaluated its performance with ROUGE-1 Recall vs. state-of-the-art extractive summarization approaches. Our results show that MUSE performs better than the best known multilingual approach (TextRank[1]) in both languages. Moreover, our experimental results on a bilingual (English and Hebrew) document collection suggest that MUSE does not need to be retrained on each language and the same model can be used across at least two different languages.

1 Introduction

Document summaries should use a minimum number of words to express a document's main ideas. As such, high quality summaries can significantly reduce the information overload many professionals in a variety of fields must contend with on a daily basis (Filippova et al., 2009), assist in the automated classification and filtering of documents, and increase search engines precision.

Automated summarization methods can use different levels of linguistic analysis: *morphological*, *syntactic*, *semantic* and *discourse/pragmatic* (Mani, 2001). Although the summary quality is expected to improve when a summarization technique includes language specific knowledge, the inclusion of that knowledge impedes the use of the summarizer on multiple languages. Only systems that perform equally well on different languages without language-specific knowledge (including linguistic analysis) can be considered language-independent summarizers.

The publication of information on the Internet in an ever-increasing variety of languages [2] dictates the importance of developing multilingual summarization approaches. There is a particular need for language-independent statistical techniques that can be readily applied to text in any language without depending on language-specific linguistic tools. In the absence of such techniques, the only alternative to language-independent summarization would be the labor-intensive translation of the entire document into a common language.

Here we introduce MUSE (MUltilingual Sentence Extractor), a new approach to multilingual single-document extractive summarization where summarization is considered as an optimization or a search problem. We use a Genetic Algorithm (GA) to find an optimal weighted linear combination of 31 statistical sentence scoring methods that are all language-independent and are based on either a vector or a graph representation of a document, where both representations are based on a

[1] We evaluated several summarizers—SUMMA, MEAD, Microsoft Word Autosummarize and TextRank—on the DUC 2002 corpus. Our results show that TextRank performed best. In addition, TextRank can be considered language-independent as long as it does not perform any morphological analysis.

[2] Gulli and Signorini (2005) used Web searches in 75 different languages to estimate the size of the Web as of the end of January 2005.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 927–936,
Uppsala, Sweden, 11-16 July 2010.

word segmentation.

We have evaluated our approach on two monolingual corpora of English and Hebrew documents and, additionally, on one bilingual corpora comprising English and Hebrew documents. Our evaluation experiments sought to
- Compare the GA-based approach for single-document extractive summarization (MUSE) to the best known sentence scoring methods.
- Determine whether the same weighting model is applicable across two different languages.

This paper is organized as follows. The next section describes the related work in statistical extractive summarization. Section 3 introduces MUSE, the GA-based approach to multilingual single-document extractive summarization. Section 4 presents our experimental results on monolingual and bilingual corpora. Our conclusions and suggestions for future work comprise the final section.

2 Related Work

Extractive summarization is aimed at the selection of a subset of the most relevant fragments from a source text into the summary. The fragments can be paragraphs (Salton et al., 1997), sentences (Luhn, 1958), keyphrases (Turney, 2000) or keywords (Litvak and Last, 2008). Statistical methods for calculating the relevance score of each fragment can be categorized into several classes: *cue*-based (Edmundson, 1969), *keyword*- or *frequency*-based (Luhn, 1958; Edmundson, 1969; Neto et al., 2000; Steinberger and Jezek, 2004; Kallel et al., 2004; Vanderwende et al., 2007), *title*-based (Edmundson, 1969; Teufel and Moens, 1997), *position*-based (Baxendale, 1958; Edmundson, 1969; Lin and Hovy, 1997; Satoshi et al., 2001) and *length*-based (Satoshi et al., 2001).

Considered the first work on sentence scoring for automated text summarization, Luhn (1958) based the significance factor of a sentence on the frequency and the relative positions of significant words within a sentence. Edmundson (1969) tested different linear combinations of four sentence ranking scoring methods—*cue*, *key*, *title* and *position*—to identify that which performed best on a training corpus. Linear combinations of several statistical sentence ranking methods were also applied in the MEAD (Radev et al., 2001) and SUMMA (Saggion et al., 2003) approaches, both of which use the vector space model for text representation and a set of predefined or user-specified weights for a combination of *position*, *frequency*, *title*, and *centroid*-based (MEAD) features. Goldstein et al. (1999) integrated linguistic and statistical features. In none of these works, however, did the researchers attempt to find the optimal weights for the best linear combination.

Information retrieval and machine learning techniques were integrated to determine sentence importance (Kupiec et al., 1995; Wong et al., 2008). Gong and Liu (2001) and Steinberger and Jezek (2004) used singular value decomposition (SVD) to generate extracts. Ishikawa et al. (2002) combined conventional sentence extraction and a trainable classifier based on support vector machines.

Some authors reduced the summarization process to an optimization or a search problem. Hassel and Sjobergh (2006) used a standard hill-climbing algorithm to build summaries that maximize the score for the total impact of the summary. A summary consists of first sentences from the document was used as a starting point for the search, and all neighbours (summaries that can be created by simply removing one sentence and adding another) were examined, looking for a better summary.

Kallel et al. (2004) and Liu et al. (2006b) used genetic algorithms (GAs), which are known as prominent search and optimization methods (Goldberg, 1989), to find sets of sentences that maximize summary quality metrics, starting from a random selection of sentences as the initial population. In this capacity, however, the high computational complexity of GAs is a disadvantage. To choose the best summary, multiple candidates should be generated and evaluated for each document (or document cluster).

Following a different approach, Turney (2000) used a GA to learn an optimized set of parameters for a keyword extractor embedded in the Extractor tool.[3] Orăsan et al. (2000) enhanced the preference-based anaphora resolution algorithms by using a GA to find an optimal set of values for the outcomes of 14 indicators and apply the optimal combination of values from data on one text to a different text. With such approach, training may be the only time-consuming phase in the operation.

[3] `http://www.extractor.com/`

Today, graph-based text representations are becoming increasingly popular, due to their ability to enrich the document model with syntactic and semantic relations. Salton et al. (1997) were among the first to make an attempt at using graph-based ranking methods in single document extractive summarization, generating similarity links between document paragraphs and using degree scores in order to extract the important paragraphs from the text. Erkan and Radev (2004) and Mihalcea (2005) introduced algorithms for unsupervised extractive summarization that rely on the application of iterative graph-based ranking algorithms, such as PageRank (Brin and Page, 1998) and HITS (Kleinberg, 1999). Their methods represent a document as a graph of sentences interconnected by similarity relations. Various similarity functions can be applied: cosine similarity as in (Erkan and Radev, 2004), simple overlap as in (Mihalcea, 2005), or other functions. Edges representing the similarity relations can be weighted (Mihalcea, 2005) or unweighted (Erkan and Radev, 2004): two sentences are connected if their similarity is above some predefined threshold value.

3 MUSE – MUltilingual Sentence Extractor

In this paper we propose a learning approach to language-independent extractive summarization where the best set of weights for a linear combination of sentence scoring methods is found by a genetic algorithm trained on a collection of document summaries. The weighting vector thus obtained is used for sentence scoring in future summarizations. Since most sentence scoring methods have a linear computational complexity, only the training phase of our approach is time-consuming.

3.1 Sentence scoring methods

Our work is aimed at identifying the best linear combination of the 31 sentence scoring methods listed in Table 1. Each method description includes a reference to the original work where the method was proposed for extractive summarization. Methods proposed in this paper are denoted by **new**. Formulas incorporate the following notation: a sentence is denoted by S, a text document by D, the total number of words in S by N, the total number of sentences in D by n, the sequential number of S in D by i, and the in-document term frequency of the term t by $tf(t)$. In the *LUHN* method, W_i and N_i are the number of keywords and the total number of words in the i^{th} cluster, respectively, such that clusters are portions of a sentence bracketed by keywords, i.e., frequent, non-common words.[4]

Figure 1 demonstrates the taxonomy of the methods listed in Table 1. Methods that require pre-defined threshold values are marked with a cross and listed in Table 2 together with the average threshold values obtained after method evaluation on English and Hebrew corpora. Each method was evaluated on both corpora, with different threshold $t \in [0, 1]$ (only numbers with one decimal digit were considered). Threshold values resulted in the best ROUGE-1 scores, were selected. A threshold of 1 means that all terms are considered, while a value of 0 means that only terms with the highest rank (*tf*, *degree*, or *pagerank*) are considered. The methods are divided into three main categories—*structure-*, *vector-*, and *graph*-based—according to the text representation model, and each category is divided into sub-categories.

Section 3.3 describes our application of a GA to the summarization task.

Table 2: Selected thresholds for threshold-based scoring methods

Method	Threshold
LUHN	0.9
LUHN_DEG	0.9
LUHN_PR	0.0
KEY	[0.8, 1.0]
KEY_DEG	[0.8, 1.0]
KEY_PR	[0.1, 1.0]
COV	0.9
COV_DEG	[0.7, 0.9]
COV_PR	0.1

3.2 Text representation models

The vector-based scoring methods listed in Table 1 use *tf* or *tf-idf* term weights to evaluate sentence importance. In contrast, representation used by the graph-based methods (except for TextRank) is based on the word-based graph representation models described in (Schenker et al., 2004). Schenker et al. (2005) showed that such graph representations can outperform the vector space model on several document categorization tasks. In the graph representation used by us in this work

[4] Luhn's experiments suggest an optimal limit of 4 or 5 non-significant words between keywords.

Table 1: Sentence scoring metrics

Name	Description	Source
POS_F	Closeness to the beginning of the document: $\frac{1}{i}$	(Edmundson, 1969)
POS_L	Closeness to the end of the document: i	(Baxendale, 1958)
POS_B	Closeness to the borders of the document: $max(\frac{1}{i}, \frac{1}{n-i+1})$	(Lin and Hovy, 1997)
LEN_W	Number of *words* in the sentence	(Satoshi et al., 2001)
LEN_CH	Number of *characters* in the sentence[5]	
LUHN	$max_{i \in \{clusters(S)\}}\{CS_i\}, CS_i = \frac{W_i^2}{N_i}$	(Luhn, 1958)
KEY	Sum of the keywords frequencies: $\sum_{t \in \{Keywords(S)\}} tf(t)$	(Edmundson, 1969)
COV	Ratio of keywords number (Coverage): $\frac{\lvert Keywords(S)\rvert}{\lvert Keywords(D)\rvert}$	(Liu et al., 2006a)
TF	Average term frequency for all sentence words: $\frac{\sum_{t \in S} tf(t)}{N}$	(Vanderwende et al., 2007)
TFISF	$\sum_{t \in S} tf(t) \times isf(t), isf(t) = 1 - \frac{log(n(t))}{log(n)}$, $n(t)$ is the number of sentences containing t	(Neto et al., 2000)
SVD	Length of a sentence vector in $\Sigma^2 \cdot V^T$ after computing Singular Value Decomposition of a term by sentences matrix $A = U\Sigma V^T$	(Steinberger and Jezek, 2004)
TITLE_O	Overlap similarity[6] to the title: $sim(S,T) = \frac{\lvert S \cap T\rvert}{min\{\lvert S\rvert,\lvert T\rvert\}}$	(Edmundson, 1969)
TITLE_J	Jaccard similarity to the title: $sim(S,T) = \frac{\lvert S \cap T\rvert}{\lvert S \cup T\rvert}$	
TITLE_C	Cosine similarity to the title: $sim(\vec{S},\vec{T}) = cos(\vec{S},\vec{T}) = \frac{\vec{S} \bullet \vec{T}}{\lvert\vec{S}\rvert \bullet \lvert\vec{T}\rvert}$	
D_COV_O	Overlap similarity to the document complement $sim(S, D-S) = \frac{\lvert S \cap T\rvert}{min\{\lvert S\rvert,\lvert D-S\rvert\}}$	**new**
D_COV_J	Jaccard similarity to the document complement $sim(S, D-S) = \frac{\lvert S \cap T\rvert}{\lvert S \cup D-S\rvert}$	
D_COV_C	Cosine similarity to the document complement $cos(\vec{S}, \vec{D-S}) = \frac{\vec{S} \bullet \vec{D-S}}{\lvert\vec{S}\rvert \bullet \lvert\vec{D-S}\rvert}$	
LUHN_DEG	Graph-based extensions of LUHN, KEY and COV measures respectively.	
KEY_DEG	Node degree is used instead of a word frequency: words are considered	
COV_DEG	significant if they are represented by nodes having a degree higher than a predefined threshold	
DEG	Average degree for all sentence nodes: $\frac{\sum_{i \in \{words(S)\}} Deg_i}{N}$	
GRASE	Frequent sentences from *bushy* paths are selected. Each sentence in the *bushy* path gets a domination score that is the number of edges with its label in the path normalized by the sentence length. The relevance score for a sentence is calculated as a sum of its domination scores over all paths.	
LUHN_PR	Graph-based extensions of LUHN, KEY and COV measures respectively.	
KEY_PR	Node PageRank score is used instead of a word frequency: words are considered	
COV_PR	significant if they are represented by nodes having a PageRank score higher than a predefined threshold	
PR	Average PageRank for all sentence nodes: $\frac{\sum_{t \in S} PR(t)}{N}$	
TITLE_E_O	Overlap-based edge matching between title and sentence graphs	
TITLE_E_J	Jaccard-based edge matching between title and sentence graphs	
D_COV_E_O	Overlap-based edge matching between sentence and a document complement graphs	
D_COV_E_J	Jaccard-based edge matching between sentence and a document complement graphs	
ML_TR	Multilingual version of TextRank without morphological analysis: Sentence score equals to PageRank (Brin and Page, 1998) rank of its node: $WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$	(Mihalcea, 2005)

nodes represent unique terms (distinct words) and edges represent order-relationships between two terms. There is a directed edge from A to B if an A term immediately precedes the B term in any sentence of the document. We label each edge with the IDs of sentences that contain both words in the specified order.

3.3 Optimization—learning the best linear combination

We found the best linear combination of the methods listed in Table 1 using a Genetic Algorithm (GA). GAs are categorized as global search heuristics. Figure 2 shows a simplified GA flowchart. A typical genetic algorithm requires (1) a genetic representation of the solution domain, and (2) a fitness function to evaluate the solution domain.

We represent the solution as a vector of weights

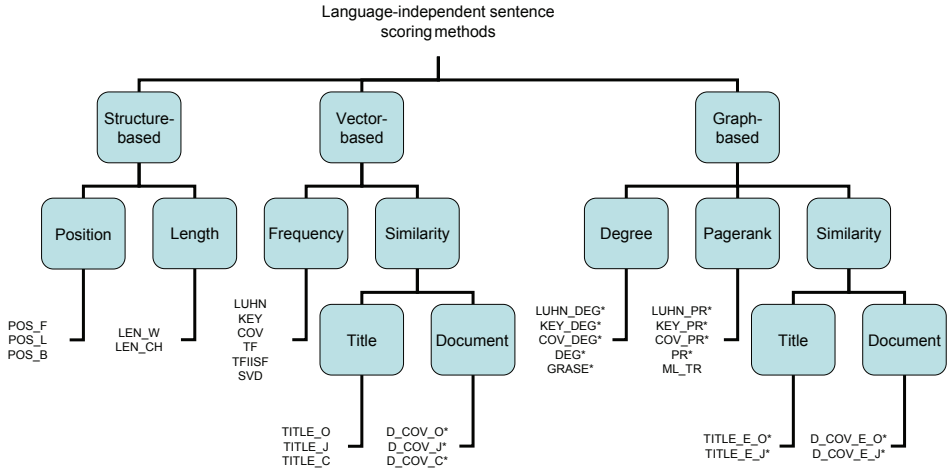


Figure 1: Taxonomy of language-independent sentence scoring methods

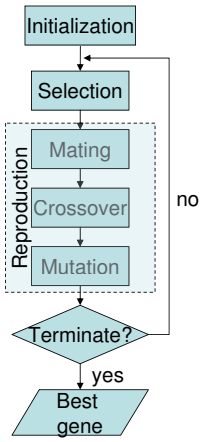


Figure 2: Simplified flowchart of a Genetic Algorithm

for a linear combination of sentence scoring methods—real-valued numbers in the unlimited range normalized in such a way that they sum up to 1. The vector size is fixed and it equals to the number of methods used in the combination.

Defined over the genetic representation, the fitness function measures the quality of the represented solution. We use ROUGE-1 Recall (Lin and Hovy, 2003) as a fitness function for measuring summarization quality, which is maximized during the optimization procedure.

Below we describe each phase of the optimization procedure in detail.

Initialization GA will explore only a small part of the search space, if the population is too small, whereas it slows down if there are too many solutions. We start from $N = 500$ randomly generated genes/solutions as an initial population, that empirically was proven as a good choice. Each gene is represented by a weighting vector $v_i = w_1, \ldots, w_D$ having a fixed number of $D \leq 31$ elements. All elements are generated from a standard normal distribution, with $\mu = 0$ and $\sigma^2 = 1$, and normalized to sum up to 1. For this solution representation, a negative weight, if it occurs, can be considered as a "penalty" for the associated metric.

Selection During each successive generation, a proportion of the existing population is selected to breed a new generation. We use a truncation selection method that rates the fitness of each solution and selects the best fifth (100 out of 500) of the individual solutions, i.e., getting the maximal ROUGE value. In such manner, we discard "bad" solutions and prevent them from reproduction. Also, we use *elitism*—method that prevents losing the best found solution in the population by copying it to the next generation.

Reproduction In this stage, new genes/solutions are introduced into the population, i.e., new points in the search space are explored. These new solutions are generated from those selected through the following genetic operators: *mating*, *crossover*, and *mutation*.

In *mating*, a pair of "parent" solutions is randomly selected, and a new solution is created using *crossover* and *mutation*, that are the most important part of a genetic algorithm. The GA performance is influenced mainly by these two operators. New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size N is generated.

Crossover is performed under the assumption

that new solutions can be improved by re-using the good parts of old solutions. However it is good to keep some part of population from one generation to the next. Our *crossover* operator includes a probability (80%) that a new and different offspring solution will be generated by calculating the weighted average of two "parent" vectors according to (Vignaux and Michalewicz, 1991). Formally, a new vector v will be created from two vectors v_1 and v_2 according to the formula $v = \lambda * v_1 + (1 - \lambda) * v_2$ (we set $\lambda = 0.5$). There is a probability of 20% that the offspring will be a duplicate of one of its parents.

Mutation in GAs functions both to preserve the existing diversity and to introduce new variation. It is aimed at preventing GA from falling into local extreme, but it should not be applied too often, because then GA will in fact change to random search. Our mutation operator includes a probability (3%) that an arbitrary weight in a vector will be changed by a uniformly randomized factor in the range of $[-0.3, 0.3]$ from its original value.

Termination The generational process is repeated until a termination condition—a plateau of solution/combination fitness such that successive iterations no longer produce better results—has been reached. The minimal improvement in our experiments was set to $\epsilon = 1.0E - 21$.

4 Experiments

4.1 Overview

The MUSE summarization approach was evaluated using a comparative experiment on two monolingual corpora of English and Hebrew texts and on a bilingual corpus of texts in both languages. We intentionally chose English and Hebrew, which belong to distinct language families (Indo-European and Semitic languages, respectfully), to ensure that the results of our evaluation would be widely generalizable. The specific goals of the experiment are to:

- Evaluate the optimal sentence scoring models induced from the corpora of summarized documents in two different languages.
- Compare the performance of the GA-based multilingual summarization method proposed in this work to the state-of-the-art approaches.
- Compare method performance on both languages.
- Determine whether the same sentence scoring model can be efficiently used for extractive summarization across two different languages.

4.2 Text preprocessing

Crucial to extractive summarization, proper sentence segmentation contributes to the quality of summarization results. For English sentences, we used the sentence splitter provided with the MEAD summarizer (Radev et al., 2001). A simple splitter that can split the text at periods, exclamation points, or question marks was used for the Hebrew text.[7]

4.3 Experiment design

The English text material we used in our experiments comprised the corpus of summarized documents available to the single document summarization task at the Document Understanding Conference, 2002 (DUC, 2002). This benchmark dataset contains 533 news articles, each accompanied by two to three human-generated abstracts of approximately 100 words each.

For the Hebrew language, however, to the best of our knowledge, no summarization benchmarks exist. To generate a corpus of summarized Hebrew texts, therefore, we set up an experiment where human assessors were given 50 news articles of 250 to 830 words each from the Website of the *Haaretz* newspaper.[8] All assessors were provided with the *Tool Assisting Human Assessors* (TAHA) software tool[9] that enables sentences to be easily selected and stored for later inclusion in the document extract. In total, 70 undergraduate students from the Department of Information Systems Engineering, Ben Gurion University of the Negev participated in the experiment. Each student participant was randomly assigned ten different documents and instructed to (1) spend at least five minutes on each document, (2) ignore dialogs and quotations, (3) read the whole document before beginning sentence extraction, (4) ignore redundant, repetitive, and overly detailed information, and (5) remain within the minimal and maximal summary length constraints (95 and 100 words, respectively). Summaries were assessed for quality by comparing each student's summary to those of all the other students using the ROUGE evalua-

[7] Although the same set of splitting rules may be used for many different languages, separate splitters were used for English and Hebrew because the MEAD splitter tool is restricted to European languages.

[8] `http://www.haaretz.co.il`

[9] TAHA can be provided upon request

tion toolkit adapted to Hebrew[10] and the ROUGE-1 metric (Lin and Hovy, 2003). We filtered all the summaries produced by assessors that received average ROUGE score below 0.5, i. e. agreed with the rest of assessors in less than 50% of cases. Finally, our corpus of summarized Hebrew texts was compiled from the summaries of about 60% of the most consistent assessors, with an average of seven extracts per single document[11]. The ROUGE scores of the selected assessors are distributed between 50 and 57 percents.

The third, bilingual, experimental corpus was assembled from documents in both languages.

4.4 Experimental Results

We evaluated English and Hebrew summaries using ROUGE-$1, 2, 3, 4, L, SU$ and W metrics, described in (2004). In agreement with Lin's (2004) conclusion, our results for the different metrics were not statistically distinguishable. However, ROUGE-1 showed the largest variation across the methods. In the following comparisons, all results are presented in terms of the ROUGE-1 Recall metric.

We estimated the ROUGE metric using 10-fold cross validation. The results of training and testing comprise the average ROUGE values obtained for English, Hebrew, and bilingual corpora (Table 3). Since we experimented with a different number of English and Hebrew documents (533 and 50, respectively), we have created 10 balanced bilingual corpora, each with the same number of English and Hebrew documents, by combining approximately 50 randomly selected English documents with all 50 Hebrew documents. Each corpus was then subjected to 10-fold cross validation, and the average results for training and testing were calculated.

We compared our approach (1) with a multilingual version of TextRank (denoted by ML_TR) (Mihalcea, 2005) as the best known multilingual summarizer, (2) with Microsoft Word's Autosummarize function[12] (denoted by MS_SUM) as a widely used commercial summarizer, and (3) with the best single scoring method in each corpus. As a baseline, we compiled summaries created from the initial sentences (denoted by POS_F). Table 4 shows the comparative results (ROUGE mean values) for English, Hebrew, and bilingual corpora, with the best summarizers on top. Pairwise comparisons between summarizers indicated that all methods (except POS_F and ML_TR in the English and bilingual corpora and D_COV_J and POS_F in the Hebrew corpus) were significantly different at the 95% confidence level. MUSE performed significantly better than TextRank in all three corpora and better than the best single methods COV_DEG in English and D_COV_J in Hebrew corpora respectively.

Two sets of features—the full set of 31 sentence scoring metrics and the 10 best bilingual metrics determined in our previous work[13] using a clustering analysis of the methods results on both corpora—were tested on the bilingual corpus. The experimental results show that the optimized combination of the 10 best metrics is not significantly distinguishable from the best single metric in the multilingual corpus – COV_DEG. The difference between the combination of all 31 metrics and COV_DEG is significant only with a one-tailed p-value of 0.0798 (considered not very significant). Both combinations significantly outperformed all the other summarizers that were compared. Table 4 contains the results of MUSE-trained weights for all 31 metrics.

Our experiments showed that the removal of highly-correlated metrics (the metric with the lower ROUGE value out of each pair of highly-correlated metrics) from the linear combination slightly improved summarization quality, but the improvement was not statistically significant. Discarding bottom ranked features (up to 50%), also, did not affect the results significantly.

Table 5 shows the best vectors generated from training MUSE on all the documents in the English, Hebrew, and multilingual (one of 10 balanced) corpora and their ROUGE training scores and number of GA iterations.

While the optimal values of the weights are expected to be nonnegative, among the actual results are some negative values. Although there is no simple explanation for this outcome, it may be related to a well-known phenomenon from Numerical Analysis called *over-relaxation* (Friedman

[10]The regular expressions specifying "word" were adapted to Hebrew alphabet. The same toolkit was used for summaries evaluation on Hebrew corpus.

[11]Dataset is available at `http://www.cs.bgu.ac.il/~litvakm/research/`

[12]We reported the following bug to Microsoft: Microsoft Word's Document.Autosummarize Method returns different results from the output of the AutoSummarize Dialog Box. In our experiments, the Method results were used.

[13]submitted to publication

and Kandel, 1994). For example, Laplace equation $\phi_{xx} + \phi_{yy} = 0$ is iteratively solved over a grid of points as follows: At each grid point let $\phi^{(n)}, \overline{\phi}^{(n)}$ denote the n^{th} iteration as calculated from the differential equation and its *modified* final value, respectively. The final value is chosen as $\omega\phi^{(n)} + (1-\omega)\overline{\phi}^{(n-1)}$. While the sum of the two weights is obviously 1, the *optimal* value of ω, which minimizes the number of iterations needed for convergence, usually satisfies $1 < \omega < 2$ (i.e., the second weight $1-\omega$ is negative) and approaches 2 the finer the grid gets. Though somewhat unexpected, this surprising result can be rigorously proved (Varga, 1962).

Table 3: Results of 10-fold cross validation

	ENG	HEB	MULT
Train	0.4483	0.5993	0.5205
Test	0.4461	0.5936	0.5027

Table 4: Summarization performance. Mean ROUGE-1

Metric	ENG	HEB	MULT
MUSE	**0.4461**	**0.5921**	**0.4633**
COV_DEG	0.4363	0.5679	0.4588
D_COV_J	0.4251	0.5748	0.4512
POS_F	0.4190	0.5678	0.4440
ML_TR	0.4138	0.5190	0.4288
MS_SUM	0.3097	0.4114	0.3184

Assuming efficient implementation, most metrics have a linear computational complexity relative to the total number of words in a document - $O(n)$. As a result, MUSE total computation time, given a trained model, is also linear (at factor of the number of metrics in a combination). The training time is proportional to the number of GA iterations multiplied by the number of individuals in a population times the fitness evaluation (ROUGE) time. On average, in our experiments the GA performed $5-6$ iterations—selection and reproduction—before reaching convergence.

5 Conclusions and future work

In this paper we introduced MUSE, a new, GA-based approach to multilingual extractive summarization. We evaluated the proposed methodology on two languages from different language families: English and Hebrew. The experimental results showed that MUSE significantly outperformed TextRank, the best known language-independent approach, in both Hebrew and English using either monolingual or bilingual corpora. Moreover, our results suggest that the same weighting model is applicable across multiple languages. In future work, one may:

- Evaluate MUSE on additional languages and language families.
- Incorporate threshold values for threshold-based methods (Table 2) into the GA-based optimization procedure.
- Improve performance of similarity-based metrics in the multilingual domain.
- Apply additional optimization techniques like Evolution Strategy (Beyer and Schwefel, 2002), which is known to perform well in a real-valued search space.
- Extend the search for the best summary to the problem of multi-object optimization, combining several summary quality metrics.

Table 5: Induced weights for the best linear combination of scoring metrics

Metric	ENG	HEB	MULT
COV_DEG	8.490	0.171	0.697
KEY_DEG	15.774	0.218	-2.108
KEY	4.734	0.471	0.346
COV_PR	-4.349	0.241	-0.462
COV	10.016	-0.112	0.865
D_COV_C	-9.499	-0.163	1.112
D_COV_J	11.337	0.710	2.814
KEY_PR	0.757	0.029	-0.326
LUHN_DEG	6.970	0.211	0.113
POS_F	6.875	0.490	0.255
LEN_CH	1.333	-0.002	0.214
LUHN	-2.253	-0.060	0.411
LUHN_PR	1.878	-0.273	-2.335
LEN_W	-13.204	-0.006	1.596
ML_TR	8.493	0.340	1.549
TITLE_E_J	-5.551	-0.060	-1.210
TITLE_E_O	-21.833	0.074	-1.537
D_COV_E_J	1.629	0.302	0.196
D_COV_O	5.531	-0.475	0.431
TFISF	-0.333	-0.503	0.232
DEG	3.584	-0.218	0.059
D_COV_E_O	8.557	-0.130	-1.071
PR	5.891	-0.639	1.793
TITLE_J	-7.551	0.071	1.445
TF	0.810	0.202	-0.650
TITLE_O	-11.996	0.179	-0.634
SVD	-0.557	0.137	0.384
TITLE_C	5.536	-0.029	0.933
POS_B	-5.350	0.347	1.074
GRASE	-2.197	-0.116	-1.655
POS_L	-22.521	-0.408	-3.531
Score	**0.4549**	**0.6019**	**0.526**
Iterations	10	6	7

Acknowledgments

We are grateful to Michael Elhadad and Galina Volk from Ben-Gurion University for providing the ROUGE toolkit adapted to the Hebrew alphabet, and to Slava Kisilevich from the University of Konstanz for the technical support in evaluation experiments.

References

P. B. Baxendale. 1958. Machine-made index for technical literaturean experiment. *IBM Journal of Research and Development*, 2(4):354–361.

H.-G. Beyer and H.-P. Schwefel. 2002. Evolution strategies: A comprehensive introduction. *Journal Natural Computing*, 1(1):3–52.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

DUC. 2002. Document understanding conference. `http://duc.nist.gov`.

H. P. Edmundson. 1969. New methods in automatic extracting. *ACM*, 16(2).

G. Erkan and D. R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

K. Filippova, M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2009. Company-oriented extractive summarization of financial news. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 246–254.

M. Friedman and A. Kandel. 1994. *Fundamentals of Computer Numerical Analysis*. CRC Press.

D. E. Goldberg. 1989. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 121–128.

Y. Gong and X. Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th ACM SIGIR conference on Research and development in information retrieval*, pages 19–25.

A. Gulli and A. Signorini. 2005. The indexable web is more than 11.5 billion pages. `http://www.cs.uiowa.edu/~asignori/web-size/`.

M. Hassel and J. Sjobergh. 2006. Towards holistic summarization: Selecting summaries, not sentences. In *Proceedings of Language Resources and Evaluation*.

K. Ishikawa, S-I. ANDO, S-I. Doi, and A. Okumura. 2002. Trainable automatic text summarization using segmentation of sentence. In *Proceedings of 2002 NTCIR 3 TSC workshop*.

F. J. Kallel, M. Jaoua, L. B. Hadrich, and A. Ben Hamadou. 2004. Summarization at laris laboratory. In *Proceedings of the Document Understanding Conference*.

J.M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

J. Kupiec, J. Pedersen, and F Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference*, pages 68–73.

C.Y. Lin and E. Hovy. 1997. Identifying topics by position. In *Proceedings of the fifth conference on Applied natural language processing*, pages 283–290.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

M. Litvak and M. Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

D. Liu, Y. He, D. Ji, and H. Yang. 2006a. Genetic algorithm based multi-document summarization. *Lecture Notes in Computer Science*, 4099:1140.

D. Liu, Y. Wang, C. Liu, and Z. Wang. 2006b. Multiple documents summarization based on genetic algorithm. *Lecture Notes in Computer Science*, 4223:355.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165.

Inderjeet Mani. 2001. *Automatic Summarization*. Natural Language Processing, John Benjamins Publishing Company.

Rada Mihalcea. 2005. Language independent extractive summarization. In *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pages 1688–1689.

J.L. Neto, A.D. Santos, C.A.A. Kaestner, and A.A. Freitas. 2000. Generating text summaries through the relative importance of topics. *Lecture Notes in Computer Science*, pages 300–309.

Constantin Orăsan, Richard Evans, and Ruslan Mitkov. 2000. Enhancing preference-based anaphora resolution with genetic algorithms. In Dimitris Christodoulakis, editor, *Proceedings of the Second International Conference on Natural Language Processing*, volume 1835, pages 185 – 195, Patras, Greece, June 2– 4. Springer.

Dragomir Radev, Sasha Blair-Goldensohn, and Zhu Zhang. 2001. Experiments in single and multidocument summarization using mead. *First Document Understanding Conference*.

Horacio Saggion, Kalina Bontcheva, and Hamish Cunningham. 2003. Robust generic and query-based summarisation. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*.

G. Salton, A. Singhal, M. Mitra, and C. Buckley. 1997. Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–207.

C. N. Satoshi, S. Satoshi, M. Murata, K. Uchimoto, M. Utiyama, and H. Isahara. 2001. Sentence extraction system assembling multiple evidence. In *Proceedings of 2nd NTCIR Workshop*, pages 319–324.

A. Schenker, H. Bunke, M. Last, and A. Kandel. 2004. Classification of web documents using graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):475–496.

A. Schenker, H. Bunke, M. Last, and A. Kandel. 2005. *Graph-theoretic techniques for web content mining*.

J. Steinberger and K. Jezek. 2004. Text summarization and singular value decomposition. *Lecture Notes in Computer Science*, pages 245–254.

S. Teufel and M. Moens. 1997. Sentence extraction as a classification task. In *Proceedings of the Workshop on Intelligent Scalable Summarization, ACL/EACL Conference*, pages 58–65.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information processing and management*, 43(6):1606–1618.

R.S. Varga. 1962. *Matrix Iterative Methods*. Prentice-Hall.

G. A. Vignaux and Z. Michalewicz. 1991. A genetic algorithm for the linear transportation problem. *IEEE Transactions on Systems, Man and Cybernetics*, 21:445–452.

K.F. Wong, M. Wu, and W. Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 985–992.

Bayesian Synchronous Tree-Substitution Grammar Induction and its Application to Sentence Compression

Elif Yamangil and **Stuart M. Shieber**
Harvard University
Cambridge, Massachusetts, USA
{elif, shieber}@seas.harvard.edu

Abstract

We describe our experiments with training algorithms for tree-to-tree synchronous tree-substitution grammar (STSG) for monolingual translation tasks such as sentence compression and paraphrasing. These translation tasks are characterized by the relative ability to commit to parallel parse trees and availability of word alignments, yet the unavailability of large-scale data, calling for a Bayesian tree-to-tree formalism. We formalize nonparametric Bayesian STSG with epsilon alignment in full generality, and provide a Gibbs sampling algorithm for posterior inference tailored to the task of *extractive sentence compression*. We achieve improvements against a number of baselines, including expectation maximization and variational Bayes training, illustrating the merits of nonparametric inference over the space of grammars as opposed to sparse parametric inference with a fixed grammar.

1 Introduction

Given an aligned corpus of tree pairs, we might want to learn a mapping between the paired trees. Such induction of tree mappings has application in a variety of natural-language-processing tasks including machine translation, paraphrase, and sentence compression. The induced tree mappings can be expressed by synchronous grammars. Where the tree pairs are isomorphic, synchronous context-free grammars (SCFG) may suffice, but in general, non-isomorphism can make the problem of rule extraction difficult (Galley and McKeown, 2007). More expressive formalisms such as synchronous tree-substitution (Eisner, 2003) or tree-adjoining grammars may better capture the pairings.

In this work, we explore techniques for inducing synchronous tree-substitution grammars (STSG) using as a testbed application extractive sentence compression. Learning an STSG from aligned trees is tantamount to determining a segmentation of the trees into elementary trees of the grammar along with an alignment of the elementary trees (see Figure 1 for an example of such a segmentation), followed by estimation of the weights for the extracted tree pairs.[1] These elementary tree pairs serve as the *rules* of the extracted grammar. For SCFG, segmentation is trivial — each parent with its immediate children is an elementary tree — but the formalism then restricts us to deriving isomorphic tree pairs. STSG is much more expressive, especially if we allow some elementary trees on the source or target side to be unsynchronized, so that insertions and deletions can be modeled, but the segmentation and alignment problems become nontrivial.

Previous approaches to this problem have treated the two steps — grammar extraction and weight estimation — with a variety of methods. One approach is to use word alignments (where these can be reliably estimated, as in our testbed application) to align subtrees and extract rules (Och and Ney, 2004; Galley et al., 2004) but this leaves open the question of finding the right level of generality of the rules — how deep the rules should be and how much lexicalization they should involve — necessitating resorting to heuristics such as *minimality* of rules, and leading to

[1] Throughout the paper we will use the word STSG to refer to the tree-to-tree version of the formalism, although the string-to-tree version is also commonly used.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 937–947,
Uppsala, Sweden, 11-16 July 2010.

large grammars. Once a given set of rules is extracted, weights can be imputed using a discriminative approach to maximize the (joint or conditional) likelihood or the classification margin in the training data (taking or not taking into account the derivational ambiguity). This option leverages a large amount of manual domain knowledge engineering and is not in general amenable to latent variable problems.

A simpler alternative to this two step approach is to use a generative model of synchronous derivation and simultaneously segment and weight the elementary tree pairs to maximize the probability of the training data under that model; the simplest exemplar of this approach uses expectation maximization (EM) (Dempster et al., 1977). This approach has two frailties. First, EM search over the space of all possible rules is computationally impractical. Second, even if such a search were practical, the method is degenerate, pushing the probability mass towards larger rules in order to better approximate the empirical distribution of the data (Goldwater et al., 2006; DeNero et al., 2006). Indeed, the optimal grammar would be one in which each tree pair in the training data is its own rule. Therefore, proposals for using EM for this task start with a precomputed subset of rules, and with EM used just to assign weights within this grammar. In summary, previous methods suffer from problems of *narrowness* of search, having to restrict the space of possible rules, and *overfitting* in preferring overly specific grammars.

We pursue the use of hierarchical probabilistic models incorporating sparse priors to simultaneously solve both the narrowness and overfitting problems. Such models have been used as generative solutions to several other segmentation problems, ranging from word segmentation (Goldwater et al., 2006), to parsing (Cohn et al., 2009; Post and Gildea, 2009) and machine translation (DeNero et al., 2008; Cohn and Blunsom, 2009; Liu and Gildea, 2009). Segmentation is achieved by introducing a prior bias towards grammars that are compact representations of the data, namely by enforcing *simplicity* and *sparsity*: preferring simple rules (smaller segments) unless the use of a complex rule is evidenced by the data (through repetition), and thus mitigating the overfitting problem. A Dirichlet process (DP) prior is typically used to achieve this interplay. Interestingly, sampling-based nonparametric inference further allows the possibility of searching over the infinite space of grammars (and, in machine translation, possible word alignments), thus side-stepping the narrowness problem outlined above as well.

In this work, we use an extension of the aforementioned models of generative segmentation for STSG induction, and describe an algorithm for posterior inference under this model that is tailored to the task of extractive sentence compression. This task is characterized by the availability of word alignments, providing a clean testbed for investigating the effects of grammar extraction. We achieve substantial improvements against a number of baselines including EM, support vector machine (SVM) based discriminative training, and variational Bayes (VB). By comparing our method to a range of other methods that are subject differentially to the two problems, we can show that both play an important role in performance limitations, and that our method helps address both as well. Our results are thus not only encouraging for grammar estimation using sparse priors but also illustrate the merits of nonparametric inference over the space of grammars as opposed to sparse parametric inference with a fixed grammar.

In the following, we define the task of extractive sentence compression and the Bayesian STSG model, and algorithms we used for inference and prediction. We then describe the experiments in extractive sentence compression and present our results in contrast with alternative algorithms. We conclude by giving examples of compression patterns learned by the Bayesian method.

2 Sentence compression

Sentence compression is the task of summarizing a sentence while retaining most of the informational content and remaining grammatical (Jing, 2000). In *extractive* sentence compression, which we focus on in this paper, an order-preserving subset of the words in the sentence are selected to form the summary, that is, we summarize by deleting words (Knight and Marcu, 2002). An example sentence pair, which we use as a running example, is the following:

- Like FaceLift, much of ATM's screen performance depends on the underlying application.
- ATM's screen performance depends on the underlying application.

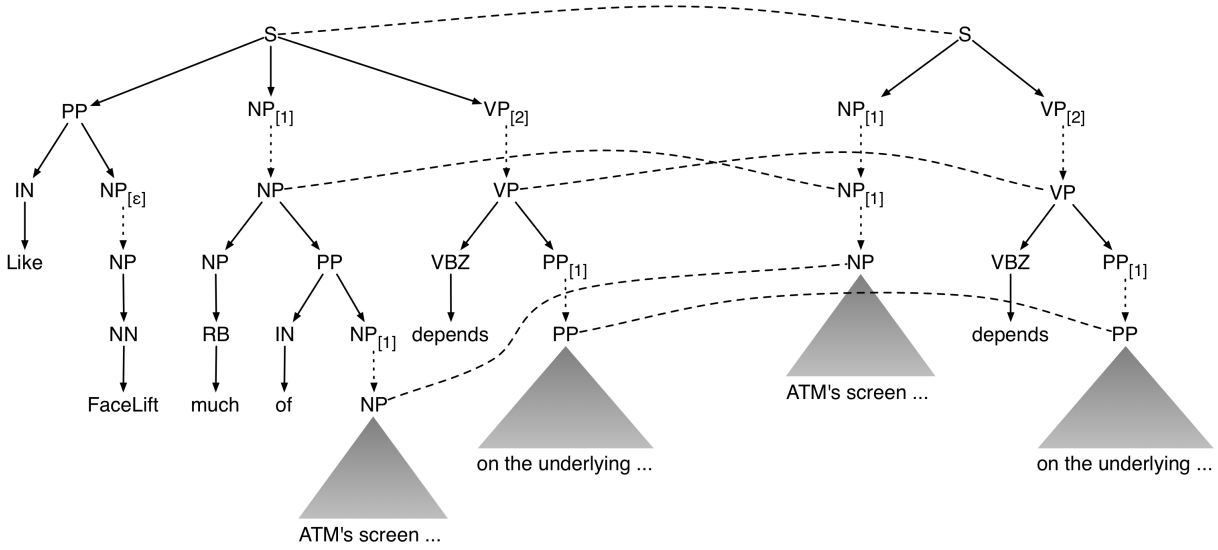


Figure 1: A portion of an STSG derivation of the example sentence and its extractive compression.

where the underlined words were deleted. In *supervised* sentence compression, the goal is to generalize from a parallel training corpus of sentences (source) and their compressions (target) to unseen sentences in a test set to predict their compressions. An *unsupervised* setup also exists; methods for the unsupervised problem typically rely on language models and linguistic/discourse constraints (Clarke and Lapata, 2006a; Turner and Charniak, 2005). Because these methods rely on dynamic programming to efficiently consider hypotheses over the space of all possible compressions of a sentence, they may be harder to extend to general paraphrasing.

3 The STSG Model

Synchronous tree-substitution grammar is a formalism for synchronously generating a pair of non-isomorphic source and target trees (Eisner, 2003). Every grammar rule is a pair of elementary trees aligned at the leaf level at their frontier nodes, which we will denote using the form

$$c_s/c_t \rightarrow e_s/e_t, \quad \gamma$$

(indices s for source, t for target) where c_s, c_t are root nonterminals of the elementary trees e_s, e_t respectively and γ is a 1-to-1 correspondence between the frontier nodes in e_s and e_t. For example, the rule

$$\text{S / S} \rightarrow \text{(S (PP (IN Like) NP}_{[\epsilon]}\text{) NP}_{[1]}\ \text{VP}_{[2]}\text{) / (S NP}_{[1]}\ \text{VP}_{[2]}\text{)}$$

can be used to delete a subtree rooted at PP. We use square bracketed indices to represent the alignment γ of frontier nodes — $\text{NP}_{[1]}$ aligns with $\text{NP}_{[1]}$, $\text{VP}_{[2]}$ aligns with $\text{VP}_{[2]}$, $\text{NP}_{[\epsilon]}$ aligns with the special symbol ϵ denoting a deletion from the source tree. Symmetrically ϵ-aligned target nodes are used to represent insertions into the target tree. Similarly, the rule

$$\text{NP / } \epsilon \rightarrow \text{(NP (NN FaceLift)) / } \epsilon$$

can be used to continue deriving the deleted subtree. See Figure 1 for an example of how an STSG with these rules would operate in synchronously generating our example sentence pair.

STSG is a convenient choice of formalism for a number of reasons. First, it eliminates the isomorphism and strong independence assumptions of SCFGs. Second, the ability to have rules deeper than one level provides a principled way of modeling lexicalization, whose importance has been emphasized (Galley and McKeown, 2007; Yamangil and Nelken, 2008). Third, we may have our STSG operate on trees instead of sentences, which allows for efficient parsing algorithms, as well as providing syntactic analyses for our predictions, which is desirable for automatic evaluation purposes.

A straightforward extension of the popular EM algorithm for probabilistic context free grammars (PCFG), the inside-outside algorithm (Lari and Young, 1990), can be used to estimate the rule weights of a given unweighted STSG based on a corpus of parallel parse trees $\mathbf{t} = t_1, \ldots, t_N$ where $t_n = t_{n,s}/t_{n,t}$ for $n = 1, \ldots, N$. Similarly, an

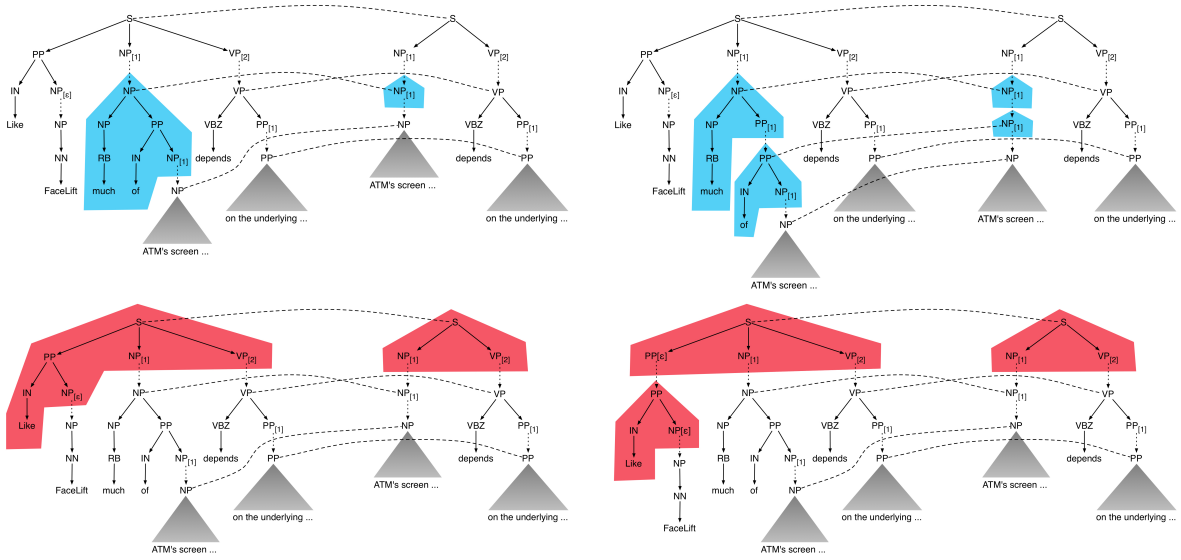


Figure 2: Gibbs sampling updates. We illustrate a sampler move to align/unalign a source node with a target node (top row in blue), and split/merge a deletion rule via aligning with ϵ (bottom row in red).

extension of the Viterbi algorithm is available for finding the maximum probability derivation, useful for predicting the target analysis $t_{N+1,t}$ for a test instance $t_{N+1,s}$. (Eisner, 2003) However, as noted earlier, EM is subject to the narrowness and overfitting problems.

3.1 The Bayesian generative process

Both of these issues can be addressed by taking a nonparametric Bayesian approach, namely, assuming that the elementary tree pairs are sampled from an independent collection of Dirichlet process (DP) priors. We describe such a process for sampling a corpus of tree pairs $\mathbf{t}$.

For all pairs of root labels $c = c_s/c_t$ that we consider, where up to one of c_s or c_t can be ϵ (e.g., S / S, NP / ϵ), we sample a sparse discrete distribution G_c over infinitely many elementary tree pairs $e = e_s/e_t$ sharing the common root c from a DP

$$G_c \sim \mathrm{DP}(\alpha_c, P_0(\cdot \mid c)) \quad (1)$$

where the DP has the concentration parameter α_c controlling the sparsity of G_c, and the base distribution $P_0(\cdot \mid c)$ is a distribution over novel elementary tree pairs that we describe more fully shortly.

We then sample a sequence of elementary tree pairs to serve as a derivation for each observed derived tree pair. For each $n = 1, \ldots, N$, we sample elementary tree pairs $e_n = e_{n,1}, \ldots, e_{n,d_n}$ in a derivation sequence (where d_n is the number of rules used in the derivation), consulting G_c whenever an elementary tree pair with root c is to be sampled.

$$e \overset{\text{iid}}{\sim} G_c, \quad \text{for all } e \text{ whose root label is } c$$

Given the derivation sequence e_n, a tree pair t_n is determined, that is,

$$p(t_n \mid e_n) = \begin{cases} 1 & e_{n,1}, \ldots, e_{n,d_n} \text{ derives } t_n \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The hyperparameters α_c can be incorporated into the generative model as random variables; however, we opt to fix these at various constants to investigate different levels of sparsity.

For the base distribution $P_0(\cdot \mid c)$ there are a variety of choices; we used the following simple scenario. (We take $c = c_s/c_t$.)

Synchronous rules For the case where neither c_s nor c_t are the special symbol ϵ, the base distribution first generates e_s and e_t independently, and then samples an alignment between the frontier nodes. Given a nonterminal, an elementary tree is generated by first making a decision to expand the nonterminal (with probability β_c) or to leave it as a frontier node ($1 - \beta_c$). If the decision to expand was made, we sample an appropriate rule from a PCFG which we estimate ahead

of time from the training corpus. We expand the nonterminal using this rule, and then repeat the same procedure for every child generated that is a nonterminal until there are no generated nonterminal children left. This is done independently for both e_s and e_t. Finally, we sample an alignment between the frontier nodes uniformly at random out of all possible alingments.

Deletion/insertion rules If $c_t = \epsilon$, that is, we have a deletion rule, we need to generate $e = e_s/\epsilon$. (The insertion rule case is symmetric.) The base distribution generates e_s using the same process described for synchronous rules above. Then with probability 1 we align all frontier nodes in e_s with ϵ. In essence, this process generates TSG rules, rather than STSG rules, which are used to cover deleted (or inserted) subtrees.

This simple base distribution does nothing to enforce an alignment between the internal nodes of e_s and e_t. One may come up with more sophisticated base distributions. However the main point of the base distribution is to encode a controllable preference towards simpler rules; we therefore make the simplest possible assumption.

3.2 Posterior inference via Gibbs sampling

Assuming fixed hyperparameters $\boldsymbol{\alpha} = \{\alpha_c\}$ and $\boldsymbol{\beta} = \{\beta_c\}$, our inference problem is to find the posterior distribution of the derivation sequences $\mathbf{e} = e_1, \ldots, e_N$ given the observations $\mathbf{t} = t_1, \ldots, t_N$. Applying Bayes' rule, we have

$$p(\mathbf{e} \mid \mathbf{t}) \quad \propto \quad p(\mathbf{t} \mid \mathbf{e}) p(\mathbf{e}) \tag{3}$$

where $p(\mathbf{t} \mid \mathbf{e})$ is a $0/1$ distribution (2) which does not depend on G_c, and $p(\mathbf{e})$ can be obtained by collapsing G_c for all c.

Consider repeatedly generating elementary tree pairs $e_1, \ldots, e_i$, all with the same root c, iid from G_c. Integrating over G_c, the e_i become dependent. The conditional prior of the i-th elementary tree pair given previously generated ones $e_{<i} = e_1, \ldots, e_{i-1}$ is given by

$$p(e_i \mid e_{<i}) \quad = \quad \frac{n_{e_i} + \alpha_c P_0(e_i \mid c)}{i - 1 + \alpha_c} \tag{4}$$

where n_{e_i} denotes the number of times e_i occurs in $e_{<i}$. Since the collapsed model is exchangeable in the e_i, this formula forms the backbone of the inference procedure that we describe next. It also makes clear DP's inductive bias to reuse elementary tree pairs.

We use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo (MCMC) method, to sample from the posterior (3). A derivation $\mathbf{e}$ of the corpus $\mathbf{t}$ is completely specified by an alignment between the source nodes and the corresponding target nodes (as well as ϵ on either side), which we take to be the state of the sampler. We start at a random derivation of the corpus, and at every iteration resample a derivation by amending the current one through local changes made at the node level, in the style of Goldwater et al. (2006).

Our sampling updates are extensions of those used by Cohn and Blunsom (2009) in MT, but are tailored to our task of extractive sentence compression. In our task, no target node can align with ϵ (which would indicate a subtree insertion), and barring unary branches no source node i can align with two different target nodes j and j' at the same time (indicating a tree expansion). Rather, the configurations of interest are those in which only source nodes i can align with ϵ, and two source nodes i and i' can align with the same target node j. Thus, the alignments of interest are not arbitrary relations, but (partial) functions from nodes in e_s to nodes in e_t or ϵ. We therefore sample in the direction from source to target. In particular, we visit every tree pair and each of its source nodes i, and update its alignment by selecting between and within two choices: (a) unaligned, (b) aligned with some target node j or ϵ. The number of possibilities j in (b) is significantly limited, firstly by the word alignment (for instance, a source node dominating a deleted subspan cannot be aligned with a target node), and secondly by the current alignment of other nearby aligned source nodes. (See Cohn and Blunsom (2009) for details of matching spans under tree constraints.)[2]

[2] One reviewer was concerned that since we explicitly disallow insertion rules in our sampling procedure, our model that generates such rules wastes probability mass and is therefore "deficient". However, we regard sampling as a separate step from the data generation process, in which we can formulate more effective algorithms by using our domain knowledge that our data set was created by annotators who were instructed to delete words only. Also, disallowing insertion rules in the base distribution unnecessarily complicates the definition of the model, whereas it is straightforward to define the joint distribution of all (potentially useful) rules and then use domain knowledge to constrain the support of that distribution during inference, as we do here. In fact, it is pos-

More formally, let e_M be the elementary tree pair rooted at the closest aligned ancestor i' of node i when it is unaligned; and let e_A and e_B be the elementary tree pairs rooted at i' and i respectively when i is aligned with some target node j or ϵ. Then, by exchangeability of the elementary trees sharing the same root label, and using (4), we have

$$p(\text{unalign}) = \frac{n_{e_M} + \alpha_{c_M} P_0(e_M \mid c_M)}{n_{c_M} + \alpha_{c_M}} \quad (5)$$

$$p(\text{align with } j) = \frac{n_{e_A} + \alpha_{c_A} P_0(e_A \mid c_A)}{n_{c_A} + \alpha_{c_A}} \quad (6)$$

$$\times \frac{n_{e_B} + \alpha_{c_B} P_0(e_B \mid c_B)}{n_{c_B} + \alpha_{c_B}} \quad (7)$$

where the counts $n_{e.}, n_{c.}$ are with respect to the current derivation of the rest of the corpus; except for n_{e_B}, n_{c_B} we also make sure to account for having generated e_A. See Figure 2 for an illustration of the sampling updates.

It is important to note that the sampler described can move from any derivation to any other derivation with positive probability (if only, for example, by virtue of fully merging and then resegmenting), which guarantees convergence to the posterior (3). However some of these transition probabilities can be extremely small due to passing through low probability states with large elementary trees; in turn, the sampling procedure is prone to local modes. In order to counteract this and to improve mixing we used simulated annealing. The probability mass function (5-7) was raised to the power $1/T$ with T dropping linearly from $T = 5$ to $T = 0$. Furthermore, using a final temperature of zero, we recover a maximum a posteriori (MAP) estimate which we denote $\mathbf{e}_{\text{MAP}}$.

3.3 Prediction

We discuss the problem of predicting a target tree $t_{N+1,t}$ that corresponds to a source tree $t_{N+1,s}$ unseen in the observed corpus $\mathbf{t}$. The maximum probability tree (MPT) can be found by considering all possible ways to derive it. However a much simpler alternative is to choose the target tree implied by the maximum probability derivation (MPD), which we define as

$$\begin{aligned} e^* &= \operatorname*{argmax}_e p(e \mid t_s, \mathbf{t}) \\ &= \operatorname*{argmax}_e \sum_{\mathbf{e}} p(e \mid t_s, \mathbf{e}) p(\mathbf{e} \mid \mathbf{t}) \end{aligned}$$

where e denotes a derivation for $t = t_s/t_t$. (We suppress the $N+1$ subscripts for brevity.) We approximate this objective first by substituting $\delta_{\mathbf{e}_{\text{MAP}}}(\mathbf{e})$ for $p(\mathbf{e} \mid \mathbf{t})$ and secondly using a finite STSG model for the infinite $p(e \mid t_s, \mathbf{e}_{\text{MAP}})$, which we obtain simply by normalizing the rule counts in $\mathbf{e}_{\text{MAP}}$. We use dynamic programming for parsing under this finite model (Eisner, 2003).[3]

Unfortunately, this approach does not ensure that the test instances are parsable, since t_s may include unseen structure or novel words. A workaround is to include all zero-count context free copy rules such as

NP / NP → (NP NP$_{[1]}$ PP$_{[2]}$) / (NP NP$_{[1]}$ PP$_{[2]}$)
NP / ϵ → (NP NP$_{[\epsilon]}$ PP$_{[\epsilon]}$) / ϵ

in order to smooth our finite model. We used Laplace smoothing (adding 1 to all counts) as it gave us interpretable results.

4 Evaluation

We compared the Gibbs sampling compressor (GS) against a version of maximum a posteriori EM (with Dirichlet parameter greater than 1) and a discriminative STSG based on SVM training (Cohn and Lapata, 2008) (SVM). EM is a natural benchmark, while SVM is also appropriate since it can be taken as the state of the art for our task.[4]

We used a publicly available extractive sentence compression corpus: the Broadcast News compressions corpus (BNC) of Clarke and Lapata (2006a). This corpus consists of 1370 sentence pairs that were manually created from transcribed Broadcast News stories. We split the pairs into training, development, and testing sets of 1000,

sible to prove that our approach is equivalent up to a rescaling of the concentration parameters. Since we fit these parameters to the data, our approach is equivalent.

[3] We experimented with MPT using Monte Carlo integration over possible derivations; the results were not significantly different from those using MPD.

[4] The comparison system described by Cohn and Lapata (2008) attempts to solve a more general problem than ours, abstractive sentence compression. However, given the nature of the data that we provided, it can only learn to compress by deleting words. Since the system is less specialized to the task, their model requires additional heuristics in decoding not needed for extractive compression, which might cause a reduction in performance. Nonetheless, because the comparison system is a generalization of the extractive SVM compressor of Cohn and Lapata (2007), we do not expect that the results would differ qualitatively.

	SVM	EM	GS
Precision	55.60	58.80	58.94
Recall	53.37	56.58	64.59
Relational F1	54.46	57.67	61.64
Compression rate	59.72	64.11	65.52

Table 1: Precision, recall, relational F1 and compression rate (%) for various systems on the 200-sentence BNC test set. The compression rate for the gold standard was 65.67%.

	SVM	EM	GS	Gold
Grammar	$2.75^{\dagger}$	2.85^{*}	3.69	4.25
Importance	2.85	2.67^{*}	3.41	3.82
Comp. rate	68.18	64.07	67.97	62.34

Table 2: Average grammar and importance scores for various systems on the 20-sentence subsample. Scores marked with $*$ are significantly different than the corresponding GS score at $\alpha < .05$ and with † at $\alpha < .01$ according to post-hoc Tukey tests. ANOVA was significant at $p < .01$ both for grammar and importance.

170, and 200 pairs, respectively. The corpus was parsed using the Stanford parser (Klein and Manning, 2003).

In our experiments with the publicly available SVM system we used all except paraphrasal rules extracted from bilingual corpora (Cohn and Lapata, 2008). The model chosen for testing had parameter for trade-off between training error and margin set to $C = 0.001$, used margin rescaling, and Hamming distance over bags of tokens with brevity penalty for loss function. EM used a subset of the rules extracted by SVM, namely all rules except non-head deleting compression rules, and was initialized uniformly. Each EM instance was characterized by two parameters: α, the smoothing parameter for MAP-EM, and δ, the smoothing parameter for augmenting the learned grammar with rules extracted from unseen data (add-$(\delta - 1)$ smoothing was used), both of which were fit to the development set using grid-search over $(1, 2]$. The model chosen for testing was $(\alpha, \delta) = (1.0001, 1.01)$.

GS was initialized at a random derivation. We sampled the alignments of the source nodes in random order. The sampler was run for 5000 iterations with annealing. All hyperparameters α_c, β_c were held constant at α, β for simplicity and were fit using grid-search over $\alpha \in [10^{-6}, 10^{6}], \beta \in [10^{-3}, 0.5]$. The model chosen for testing was $(\alpha, \beta) = (100, 0.1)$.

As an automated metric of quality, we compute F-score based on grammatical relations (relational F1, or RelF1) (Riezler et al., 2003), by which the consistency between the set of predicted grammatical relations and those from the gold standard is measured, which has been shown by Clarke and Lapata (2006b) to correlate reliably with human judgments. We also conducted a small human subjective evaluation of the grammaticality and informativeness of the compressions generated by the various methods.

4.1 Automated evaluation

For all three systems we obtained predictions for the test set and used the Stanford parser to extract grammatical relations from predicted trees and the gold standard. We computed precision, recall, RelF1 (all based on grammatical relations), and compression rate (percentage of the words that are *retained*), which we report in Table 1. The results for GS are averages over five independent runs. EM gives a strong baseline since it already uses rules that are limited in depth and number of frontier nodes by stipulation, helping with the overfitting we have mentioned, surprisingly outperforming its discriminative counterpart in both precision and recall (and consequently RelF1). GS however maintains the same level of precision as EM while improving recall, bringing an overall improvement in RelF1.

4.2 Human evaluation

We randomly subsampled our 200-sentence test set for 20 sentences to be evaluated by human judges through Amazon Mechanical Turk. We asked 15 self-reported native English speakers for their judgments of GS, EM, and SVM output sentences and the gold standard in terms of *grammaticality* (how fluent the compression is) and *importance* (how much of the meaning of and important information from the original sentence is retained) on a scale of 1 (worst) to 5 (best). We report in Table 2 the average scores. EM and SVM perform at very similar levels, which we attribute to using the same set of rules, while GS performs at a level substantially better than both, and much closer to human performance in both criteria. The

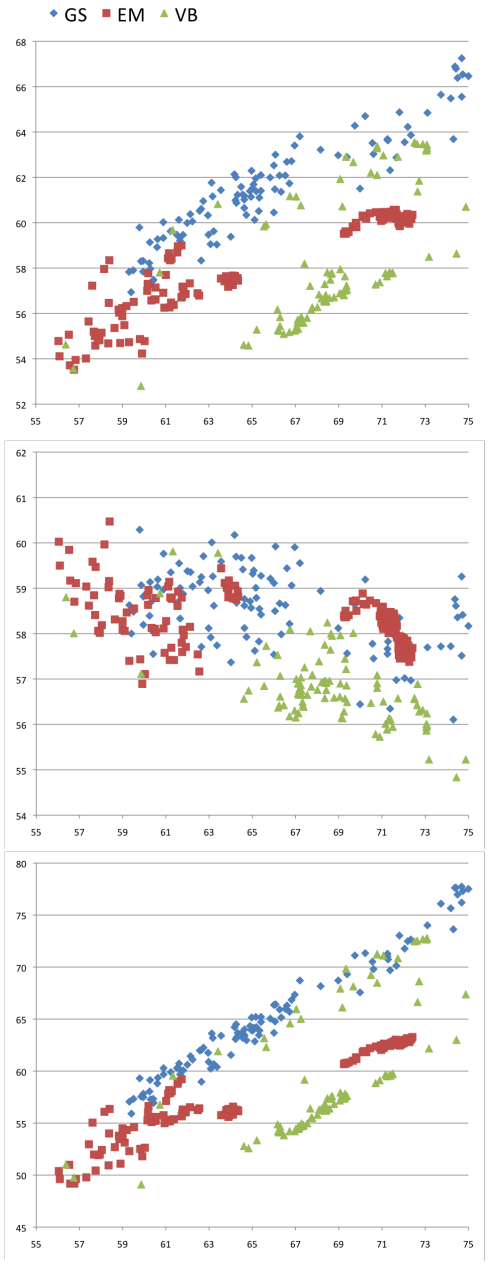


Figure 3: RelF1, precision, recall plotted against compression rate for GS, EM, VB.

human evaluation indicates that the superiority of the Bayesian nonparametric method is underappreciated by the automated evaluation metric.

4.3 Discussion

The fact that GS performs better than EM can be attributed to two reasons: (1) GS uses a sparse prior and selects a compact representation of the data (grammar sizes ranged from 4K-7K for GS compared to a grammar of about 35K rules for EM). (2) GS does not commit to a precomputed grammar and searches over the space of all grammars to find one that bests represents the corpus. It is possible to introduce DP-like sparsity in EM using variational Bayes (VB) training. We experiment with this next in order to understand how dominant the two factors are. The VB algorithm requires a simple update to the M-step formulas for EM where the expected rule counts are normalized, such that instead of updating the rule weight in the t-th iteration as in the following

$$\theta_{c,e}^{t+1} = \frac{n_{c,e} + \alpha - 1}{n_{c,\cdot} + K\alpha - K}$$

where $n_{c,e}$ represents the expected count of rule $c \rightarrow e$, and K is the total number of ways to rewrite c, we now take into account our DP$(\alpha_c, P_0(\cdot \mid c))$ prior in (1), which, when truncated to a finite grammar, reduces to a K-dimensional Dirichlet prior with parameter $\alpha_c P_0(\cdot \mid c)$. Thus in VB we perform a *variational* E-step with the subprobabilities given by

$$\theta_{c,e}^{t+1} = \frac{\exp\left(\Psi(n_{c,e} + \alpha_c P_0(e \mid c))\right)}{\exp\left(\Psi(n_{c,\cdot} + \alpha_c)\right)}$$

where Ψ denotes the digamma function. (Liu and Gildea, 2009) (See MacKay (1997) for details.) Hyperparameters were handled the same way as for GS.

Instead of selecting a single model on the development set, here we provide the whole spectrum of models and their performances in order to better understand their comparative behavior. In Figure 3 we plot RelF1 on the test set versus compression rate and compare GS, EM, and VB ($\beta = 0.1$ fixed, (α, δ) ranging in $[10^{-6}, 10^{6}] \times (1, 2]$). Overall, we see that GS maintains roughly the same level of precision as EM (despite its larger compression rates) while achieving an improvement in recall, consequently performing at a higher RelF1 level. We note that VB somewhat bridges the gap between GS and EM, without quite reaching GS performance. We conclude that the mitigation of the two factors (narrowness and overfitting) both contribute to the performance gain of GS.[5]

4.4 Example rules learned

In order to provide some insight into the grammar extracted by GS, we list in Tables (3) and (4) high

[5]We have also experimented with VB with parametric independent symmetric Dirichlet priors. The results were similar to EM with the exception of sparse priors resulting in smaller grammars and slightly improving performance.

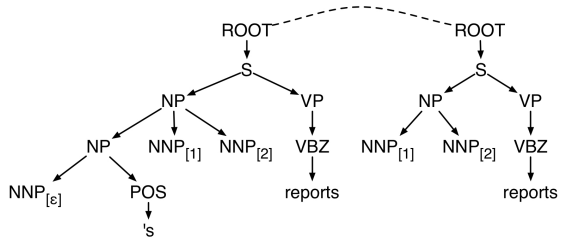
(ROOT (S CC$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S NP$_{[1]}$ ADVP$_{[\epsilon]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S ADVP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S PP$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S NP$_{[\epsilon]}$ (VP VBP$_{[\epsilon]}$ (SBAR (S NP$_{[1]}$ VP$_{[2]}$))) .$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S ADVP$_{[\epsilon]}$ NP$_{[1]}$ (VP MD$_{[2]}$ VP$_{[3]}$) .$_{[4]}$))	/ (ROOT (S NP$_{[1]}$ (VP MD$_{[2]}$ VP$_{[3]}$) .$_{[4]}$))
(ROOT (S (SBAR (IN as) S$_{[\epsilon]}$) ,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S S$_{[\epsilon]}$ (, ,) CC$_{[\epsilon]}$ (S NP$_{[1]}$ VP$_{[2]}$) .$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S PP$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$))
(ROOT (S S$_{[1]}$ (, ,) CC$_{[\epsilon]}$ S$_{[2]}$ (. .)))	/ (ROOT (S NP$_{[1]}$ VP$_{[2]}$ (. .)))
(ROOT (S S$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ ADVP$_{[2]}$ VP$_{[3]}$.$_{[4]}$))	/ (ROOT (S NP$_{[1]}$ ADVP$_{[2]}$ VP$_{[3]}$.$_{[4]}$))
(ROOT (S (NP (NP NNP$_{[\epsilon]}$ (POS 's)) NNP$_{[1]}$ NNP$_{[2]}$) (VP (VBZ reports)) .$_{[3]}$))	/ (ROOT (S (NP NNP$_{[1]}$ NNP$_{[2]}$) (VP (VBZ reports)) .$_{[3]}$))

Table 3: High probability ROOT / ROOT compression rules from the final state of the sampler.

(S NP$_{[1]}$ ADVP$_{[\epsilon]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S INTJ$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S (INTJ (UH Well)) ,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S ADVP$_{[\epsilon]}$ (, ,) S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)	/ (S S$_{[1]}$ (, ,) (CC but) S$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S NP$_{[\epsilon]}$ (VP VBP$_{[\epsilon]}$ (SBAR (IN that) (S NP$_{[1]}$ VP$_{[2]}$))) (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\epsilon]}$ (VP VBZ$_{[\epsilon]}$ ADJP$_{[\epsilon]}$ SBAR$_{[1]}$))	/ S$_{[1]}$
(S CC$_{[\epsilon]}$ PP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$ (. .))	/ (S NP$_{[1]}$ VP$_{[2]}$ (. .))
(S NP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S NP$_{[1]}$ (, ,) ADVP$_{[\epsilon]}$ (, ,) VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)
(S CC$_{[\epsilon]}$ (NP PRP$_{[1]}$) VP$_{[2]}$)	/ (S (NP PRP$_{[1]}$) VP$_{[2]}$)
(S ADVP$_{[\epsilon]}$,$_{[\epsilon]}$ PP$_{[\epsilon]}$,$_{[\epsilon]}$ NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$.$_{[3]}$)
(S ADVP$_{[\epsilon]}$ (, ,) NP$_{[1]}$ VP$_{[2]}$)	/ (S NP$_{[1]}$ VP$_{[2]}$)

Table 4: High probability S / S compression rules from the final state of the sampler.

probability subtree-deletion rules expanding categories ROOT / ROOT and S / S, respectively. Of especial interest are deep lexicalized rules such as



a pattern of compression used many times in the BNC in sentence pairs such as "NPR's Anne Garrels reports" / "Anne Garrels reports". Such an informative rule with nontrivial collocation (between the possessive marker and the word "reports") would be hard to extract heuristically and can only be extracted by reasoning across the training examples.

5 Conclusion

We explored nonparametric Bayesian learning of non-isomorphic tree mappings using Dirichlet process priors. We used the task of extractive sentence compression as a testbed to investigate the effects of sparse priors and nonparametric inference over the space of grammars. We showed that, despite its degeneracy, expectation maximization is a strong baseline when given a reasonable grammar. However, Gibbs-sampling–based nonparametric inference achieves improvements against this baseline. Our investigation with variational Bayes showed that the improvement is due both to finding sparse grammars (mitigating overfitting) and to searching over the space of all grammars (mitigating narrowness). Overall, we take these results as being encouraging for STSG induction via Bayesian nonparametrics for monolingual translation tasks. The future for this work would involve natural extensions such as mixing over the space of word alignments; this would allow application to MT-like tasks where flexible word reordering is allowed, such as abstractive sentence compression and paraphrasing.

References

James Clarke and Mirella Lapata. 2006a. Constraint-based sentence compression: An integer programming approach. In *Proceedings of the 21st Interna-*

tional Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 144–151, Sydney, Australia, July. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2006b. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384, Sydney, Australia, July. Association for Computational Linguistics.

Trevor Cohn and Phil Blunsom. 2009. A Bayesian model of syntax-directed tree to string grammar induction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 352–361, Morristown, NJ, USA. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning*, pages 73–82, Prague. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 137–144, Manchester, United Kingdom. Association for Computational Linguistics.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Morristown, NJ, USA. Association for Computational Linguistics.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 31–38, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 314–323, Morristown, NJ, USA. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 205–208, Morristown, NJ, USA. Association for Computational Linguistics.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187, Rochester, New York, April. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

S. Geman and D. Geman. 1984. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. pages 6:721–741.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia, July. Association for Computational Linguistics.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315, Morristown, NJ, USA. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS*, pages 3–10. MIT Press.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Ding Liu and Daniel Gildea. 2009. Bayesian learning of phrasal tree-to-string templates. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1308–1317, Singapore, August. Association for Computational Linguistics.

David J.C. MacKay. 1997. Ensemble learning for hidden markov models. Technical report, Cavendish Laboratory, Cambridge, UK.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449.

Matt Post and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48, Suntec, Singapore, August. Association for Computational Linguistics.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 118–125, Morristown, NJ, USA. Association for Computational Linguistics.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 290–297, Morristown, NJ, USA. Association for Computational Linguistics.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*, pages 137–140, Columbus, Ohio, June. Association for Computational Linguistics.

Contextualizing Semantic Representations Using Syntactically Enriched Vector Models

Stefan Thater and **Hagen Fürstenau** and **Manfred Pinkal**
Department of Computational Linguistics
Saarland University
{stth, hagenf, pinkal}@coli.uni-saarland.de

Abstract

We present a syntactically enriched vector model that supports the computation of contextualized semantic representations in a quasi compositional fashion. It employs a systematic combination of first- and second-order context vectors. We apply our model to two different tasks and show that (i) it substantially outperforms previous work on a paraphrase ranking task, and (ii) achieves promising results on a word-sense similarity task; to our knowledge, it is the first time that an unsupervised method has been applied to this task.

1 Introduction

In the logical paradigm of natural-language semantics originating from Montague (1973), semantic structure, composition and entailment have been modelled to an impressive degree of detail and formal consistency. These approaches, however, lack coverage and robustness, and their impact on realistic natural-language applications is limited: The logical framework suffers from over-specificity, and is inappropriate to model the pervasive vagueness, ambivalence, and uncertainty of natural-language semantics. Also, the handcrafting of resources covering the huge amounts of content which are required for deep semantic processing is highly inefficient and expensive.

Co-occurrence-based semantic vector models offer an attractive alternative. In the standard approach, word meaning is represented by feature vectors, with large sets of context words as dimensions, and their co-occurrence frequencies as values. Semantic similarity information can be acquired using unsupervised methods at virtually no cost, and the information gained is soft and gradual. Many NLP tasks have been modelled successfully using vector-based models. Examples include information retrieval (Manning et al., 2008), word-sense discrimination (Schütze, 1998) and disambiguation (McCarthy and Carroll, 2003), to name but a few.

Standard vector-space models have serious limitations, however: While semantic information is typically encoded in phrases and sentences, distributional semantics, in sharp contrast to logic-based semantics, does not offer any natural concept of compositionality that would allow the semantics of a complex expression to be computed from the meaning of its parts. A different, but related problem is caused by word-sense ambiguity and contextual variation of usage. Frequency counts of context words for a given target word provide invariant representations averaging over all different usages of the target word. There is no obvious way to distinguish the different senses of e.g. *acquire* in different contexts, such as *acquire knowledge* or *acquire shares*.

Several approaches for word-sense disambiguation in the framework of distributional semantics have been proposed in the literature (Schütze, 1998; McCarthy and Carroll, 2003). In contrast to these approaches, we present a method to model the mutual contextualization of words in a phrase in a compositional way, guided by syntactic structure. To some extent, our method resembles the approaches proposed by Mitchell and Lapata (2008) and Erk and Padó (2008). We go one step further, however, in that we employ *syntactically enriched vector models* as the basic meaning representations, assuming a vector space spanned by combinations of dependency relations and words (Lin, 1998). This allows us to model the semantic interaction between the meaning of a head word and its dependent at the micro-level of relation-specific co-occurrence frequencies. It turns out that the benefit to precision is considerable.

Using syntactically enriched vector models raises problems of different kinds: First, the use

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 948–957,
Uppsala, Sweden, 11-16 July 2010.

of syntax increases dimensionality and thus may cause data sparseness (Padó and Lapata, 2007). Second, the vectors of two syntactically related words, e.g., a target verb *acquire* and its direct object *knowledge*, typically have different syntactic environments, which implies that their vector representations encode complementary information and there is no direct way of combining the information encoded in the respective vectors.

To solve these problems, we build upon previous work (Thater et al., 2009) and propose to use *syntactic second-order vector representations*. Second-order vector representations in a bag-of-words setting were first used by Schütze (1998); in a syntactic setting, they also feature in Dligach and Palmer (2008). For the problem at hand, the use of second-order vectors alleviates the sparseness problem, and enables the definition of vector space transformations that make the distributional information attached to words in different syntactic positions compatible. Thus, it allows vectors for a predicate and its arguments to be combined in a compositional way.

We conduct two experiments to assess the suitability of our method. Our first experiment is carried out on the SemEval 2007 lexical substitution task dataset (McCarthy and Navigli, 2007). It will show that our method significantly outperforms other unsupervised methods that have been proposed in the literature to rank words with respect to their semantic similarity in a given linguistic context. In a second experiment, we apply our model to the "word sense similarity task" recently proposed by Erk and McCarthy (2009), which is a refined variant of a word-sense disambiguation task. The results show a substantial positive effect.

Plan of the paper. We will first review related work in Section 2, before presenting our model in Section 3. In Sections 4 and 5 we evaluate our model on the two different tasks. Section 6 concludes.

2 Related Work

Several approaches to contextualize vector representations of word meaning have been proposed. One common approach is to represent the meaning of a word a in context b simply as the sum, or centroid of a and b (Landauer and Dumais, 1997).

Kintsch (2001) considers a variant of this simple model. By using vector representations of a predicate p and an argument a, Kintsch identifies words that are similar to p and a, and takes the centroid of these words' vectors to be the representation of the complex expression $p(a)$.

Mitchell and Lapata (2008), henceforth M&L, propose a general framework in which meaning representations for complex expressions are computed compositionally by combining the vector representations of the individual words of the complex expression. They focus on the assessment of different operations combining the vectors of the subexpressions. An important finding is that component-wise multiplication outperforms the more common addition method. Although their composition method is guided by syntactic structure, the actual instantiations of M&L's framework are insensitive to syntactic relations and word-order, assigning identical representation to *dog bites man* and *man bites dog* (see Erk and Padó (2008) for a discussion). Also, they use syntax-free bag-of-words-based vectors as basic representations of word meaning.

Erk and Padó (2008), henceforth E&P, represent the meaning of a word w through a collection of vectors instead of a single vector: They assume selectional preferences and inverse selectional preferences to be constitutive parts of the meaning in addition to the meaning proper. The interpretation of a word p in context a is a combination of p's meaning with the (inverse) selectional preference of a. Thus, a verb meaning does not combine directly with the meaning of its object noun, as on the M&L account, but with the centroid of the vectors of the verbs to which the noun can stand in an object relation. Clearly, their approach is sensitive to syntactic structure. Their evaluation shows that their model outperforms the one proposed by M&L on a lexical substitution task (see Section 4). The basic vectors, however, are constructed in a word space similar to the one of the M&L approach.

In Thater et al. (2009), henceforth TDP, we took up the basic idea from E&P of exploiting selectional preference information for contextualization. Instead of using collections of different vectors, we incorporated syntactic information by assuming a richer internal structure of the vector representations. In a small case study, moderate improvements over E&P on a lexical substitution task could be shown. In the present paper, we formulate a general model of syntactically informed contextualization and show how to apply it to a number a of representative lexical substitution tasks. Evaluation shows significant improvements over TDP

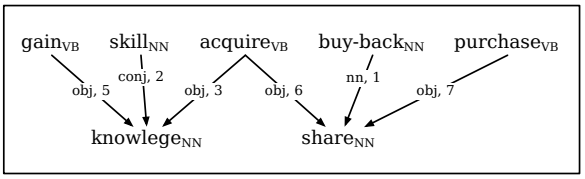


Figure 1: Co-occurrence graph of a small sample corpus of dependency trees.

and E&P.

3 The model

In this section, we present our method of contextualizing semantic vector representations. We first give an overview of the main ideas, which is followed by a technical description of first-order and second-order vectors (Section 3.2) and the contextualization operation (Section 3.3).

3.1 Overview

Our model employs vector representations for words and expressions containing syntax-specific first and second order co-occurrences information.

The basis for the construction of both kinds of vector representations are co-occurrence graphs. Figure 1 shows the co-occurrence graph of a small sample corpus of dependency trees: Words are represented as nodes in the graph, possible dependency relations between them are drawn as labeled edges, with weights corresponding to the observed frequencies. From this graph, we can directly read off the first-order vector for every word w: the vector's dimensions correspond to pairs (r, w') of a grammatical relation and a neighboring word, and are assigned the frequency count of (w, r, w').

The noun *knowledge*, for instance, would be represented by the following vector:

$$\langle 5_{(\text{OBJ}^{-1},\text{gain})}, 2_{(\text{CONJ}^{-1},\text{skill})}, 3_{(\text{OBJ}^{-1},\text{acquire})}, \ldots \rangle$$

This vector talks about the possible dependency heads of *knowledge* and thus can be seen as the (inverse) selectional preference of *knowledge* (see Erk and Padó (2008)).

As soon as we want to compute a meaning representation for a phrase like *acquire knowledge* from the verb *acquire* together with its direct object *knowledge*, we are facing the problem that verbs have different syntactic neighbors than nouns, hence their first-order vectors are not easily comparable. To solve this problem we additionally introduce another kind of vectors capturing informations about all words that can be reached with two steps in the co-occurrence graph. Such a path is characterized by two dependency relations and two words, i.e., a quadruple (r, w', r', w''), whose weight is the product of the weights of the two edges used in the path. To avoid overly sparse vectors we generalize over the "middle word" w' and build our second-order vectors on the dimensions corresponding to triples (r, r', w'') of two dependency relations and one word at the end of the two-step path. For instance, the second-order vector for *acquire* is

$$\langle 15_{(\text{OBJ},\text{OBJ}^{-1},\text{gain})}, 6_{(\text{OBJ},\text{CONJ}^{-1},\text{skill})}, 6_{(\text{OBJ},\text{OBJ}^{-1},\text{buy-back})}, 42_{(\text{OBJ},\text{OBJ}^{-1},\text{purchase})}, \ldots \rangle$$

In this simple example, the values are the products of the edge weights on each of the paths. The method of computation is detailed in Section 3.2. Note that second order vectors in particular contain paths of the form (r, r^{-1}, w'), relating a verb w to other verbs w' which are possible substitution candidates.

With first- and second-order vectors we can now model the interaction of semantic information within complex expressions. Given a pair of words in a particular grammatical relation like *acquire knowledge*, we contextualize the second-order vector of *acquire* with the first-order vector of *knowledge*. We let the first-order vector with its selectional preference information act as a kind of weighting filter on the second-order vector, and thus refine the meaning representation of the verb. The actual operation we will use is pointwise multiplication, which turned out to be the best-performing one for our purpose. Interestingly, Mitchell and Lapata (2008) came to the same result in a different setting.

In our example, we obtain a new second-order vector for *acquire* in the context of *knowledge*:

$$\langle 75_{(\text{OBJ},\text{OBJ}^{-1},\text{gain})}, 12_{(\text{OBJ},\text{CONJ}^{-1},\text{skill})}, 0_{(\text{OBJ},\text{OBJ}^{-1},\text{buy-back})}, 0_{(\text{OBJ},\text{OBJ}^{-1},\text{purchase})}, \ldots \rangle$$

Note that all dimensions that are not "licensed" by the argument *knowledge* are filtered out as they are multiplied with 0. Also, contextualisation of *acquire* with the argument *share* instead of *knowledge*

would have led to a very different vector, which reflects the fact that the two argument nouns induce different readings of the inherently ambiguous *acquire*.

3.2 First and second-order vectors

Assuming a set W of words and a set R of dependency relation labels, we consider a Euclidean vector space V_1 spanned by the set of orthonormal basis vectors $\{\vec{e}_{r,w'} \mid r \in R, w' \in W\}$, i.e., a vector space whose dimensions correspond to pairs of a relation and a word. Recall that any vector of V_1 can be represented as a finite sum of the form $\sum a_i \vec{e}_{r,w'}$ with appropriate scalar factors a_i. In this vector space we define the *first-order vector* $[w]$ of a word w as follows:

$$[w] = \sum_{\substack{r \in R \\ w' \in W}} \omega(w, r, w') \cdot \vec{e}_{r,w'}$$

where ω is a function that assigns the dependency triple (w, r, w') a corresponding weight. In the simplest case, ω would denote the frequency in a corpus of dependency trees of w occurring together with w' in relation r. In the experiments reported below, we use *pointwise mutual information* (Church and Hanks, 1990) instead as it proved superior to raw frequency counts:

$$pmi(w, r, w') = \log \frac{p(w, w' \mid r)}{p(w \mid r)p(w' \mid r)}$$

We further consider a similarly defined vector space V_2, spanned by an orthonormal basis $\{\vec{e}_{r,r',w'} \mid r, r' \in R, w' \in W\}$. Its dimensions therefore correspond to triples of two relations and a word. Evidently this is a higher dimensional space than V_1, which therefore can be embedded into V_2 by the "lifting maps" $L_r : V_1 \hookrightarrow V_2$ defined by $L_r(\vec{e}_{r',w'}) := \vec{e}_{r,r',w'}$ (and by linear extension therefore on all vectors of V_1). Using these lifting maps we define the *second-order vector* $[\![w]\!]$ of a word w as

$$[\![w]\!] = \sum_{\substack{r \in R \\ w' \in W}} \omega(w, r, w') \cdot L_r\left([w']\right)$$

Substituting the definitions of L_r and $[w']$, this yields

$$[\![w]\!] = \sum_{\substack{r, r' \in R \\ w'' \in W}} \left(\sum_{w' \in W} \omega(w, r, w')\omega(w', r', w'') \right) \vec{e}_{r,r',w''}$$

which shows the generalization over w' in form of the inner sum.

For example, if w is a verb, $r = \textsc{Obj}$ and $r' = \textsc{Obj}^{-1}$ (i.e., the inverse object relation), then the coefficients of $\vec{e}_{r,r',w''}$ in $[\![w]\!]$ would characterize the distribution of verbs w'' which share objects with w.

3.3 Composition

Both first and second-order vectors are defined for lexical expressions only. In order to represent the meaning of complex expressions we need to combine the vectors for grammatically related words in a given sentence. Given two words w and w' in relation r we contextualize the second-order vector of w with the r-lifted first-order vector of w':

$$[\![w_{r:w'}]\!] = [\![w]\!] \times L_r([w'])$$

Here $\times$ may denote any operator on V_2. The objective is to incorporate (inverse) selectional preference information from the context (r, w') in such a way as to identify the correct word sense of w. This suggests that the dimensions of $[\![w]\!]$ should be filtered so that only those compatible with the context remain. A more flexible approach than simple filtering, however, is to re-weight those dimensions with context information. This can be expressed by pointwise vector multiplication (in terms of the given basis of V_2). We therefore take $\times$ to be pointwise multiplication.

To contextualize (the vector of) a word w with multiple words $w_1, \ldots, w_n$ and corresponding relations $r_1, \ldots, r_n$, we compute the sum of the results of the pairwise contextualizations of the target vector with the vectors of the respective dependents:

$$[\![w_{r_1:w_1,\ldots,r_n:w_n}]\!] = \sum_{k=1}^{n} [\![w_{r_k:w_k}]\!]$$

4 Experiments: Ranking Paraphrases

In this section, we evaluate our model on a paraphrase ranking task. We consider sentences with an occurrence of some target word w and a list of paraphrase candidates $w_1, \ldots, w_k$ such that each of the w_i is a paraphrase of w for some sense of w. The task is to decide for each of the paraphrase candidates w_i how appropriate it is as a paraphrase of w in the given context. For instance, *buy*, *purchase* and *obtain* are all paraphrases of *acquire*, in the sense that they can be substituted for *acquire* in some contexts, but *purchase* and *buy* are not paraphrases of *acquire* in the first sentence of Table 1.

Sentence	Paraphrases
Teacher education students will *acquire* the knowledge and skills required to [...]	gain 4; amass 1; receive 1; obtain 1
Ontario Inc. will [...] *acquire* the remaining IXOS shares [...]	buy 3; purchase 1; gain 1; get 1; procure 2; obtain 1

Table 1: Two examples from the lexical substitution task data set

4.1 Resources

We use a vector model based on dependency trees obtained from parsing the English Gigaword corpus (LDC2003T05). The corpus consists of news from several newswire services, and contains over four million documents. We parse the corpus using the Stanford parser[1] (de Marneffe et al., 2006) and a non-lexicalized parser model, and extract over 1.4 billion dependency triples for about 3.9 million words (lemmas) from the parsed corpus.

To evaluate the performance of our model, we use various subsets of the SemEval 2007 lexical substitution task (McCarthy and Navigli, 2007) dataset. The complete dataset contains 10 instances for each of 200 target words—nouns, verbs, adjectives and adverbs—in different sentential contexts. Systems that participated in the task had to generate paraphrases for every instance, and were evaluated against a gold standard containing up to 10 possible paraphrases for each of the individual instances.

There are two natural subtasks in generating paraphrases: identifying paraphrase candidates and ranking them according to the context. We follow E&P and evaluate it only on the second subtask: we extract paraphrase candidates from the gold standard by pooling all annotated gold-standard paraphrases for all instances of a verb in all contexts, and use our model to rank these paraphrase candidates in specific contexts. Table 1 shows two instances of the target verb *acquire* together with its paraphrases in the gold standard as an example. The paraphrases are attached with weights, which correspond to the number of times they have been given by different annotators.

4.2 Evaluation metrics

To evaluate the performance of our method we use *generalized average precision* (Kishida, 2005), a variant of *average precision.*

Average precision (Buckley and Voorhees, 2000) is a measure commonly used to evaluate systems that return ranked lists of results. Generalized average precision (GAP) additionally rewards the correct order of positive cases w.r.t. their gold standard weight. We define average precision first:

$$AP = \frac{\Sigma_{i=1}^{n} x_i \, p_i}{R} \qquad p_i = \frac{\Sigma_{k=1}^{i} x_k}{i}$$

where x_i is a binary variable indicating whether the ith item as ranked by the model is in the gold standard or not, R is the size of the gold standard, and n is the number of paraphrase candidates to be ranked. If we take x_i to be the gold standard weight of the ith item or zero if it is not in the gold standard, we can define *generalized average precision* as follows:

$$GAP = \frac{\sum_{i=1}^{n} I(x_i) \, p_i}{\sum_{i=1}^{R} I(y_i)\bar{y}_i}$$

where $I(x_i) = 1$ if x_i is larger than zero, zero otherwise, and $\bar{y}_i$ is the average weight of the ideal ranked list $y_1, \ldots, y_i$ of gold standard paraphrases.

As a second scoring method, we use *precision out of ten* (P_{10}). The measure is less discriminative than GAP. We use it because we want to compare our model with E&P. P_{10} measures the percentage of gold-standard paraphrases in the top-ten list of paraphrases as ranked by the system, and can be defined as follows (McCarthy and Navigli, 2007):

$$P_{10} = \frac{\Sigma_{s \in M \cap G} \, f(s)}{\Sigma_{s \in G} \, f(s)},$$

where M is the list of 10 paraphrase candidates top-ranked by the model, G is the corresponding annotated gold-standard data, and $f(s)$ is the weight of the individual paraphrases.

4.3 Experiment 1: Verb paraphrases

In our first experiment, we consider verb paraphrases using the same controlled subset of the

[1] We use version 1.6 of the parser. We modify the dependency trees by "folding" prepositions into the edge labels to make the relation between a head word and the head noun of a prepositional phrase explicit.

lexical substitution task data that had been used by TDP in an earlier study. We compare our model to various baselines and the models of TDP and E&P, and show that our new model substantially outperforms previous work.

Dataset. The dataset is identical to the one used by TDP and has been constructed in the same way as the dataset used by E&P: it contains those gold-standard instances of verbs that have—according to the analyses produced by the MiniPar parser (Lin, 1993)—an overtly realized subject and object. Gold-standard paraphrases that do not occur in the parsed British National Corpus are removed.[2] In total, the dataset contains 162 instances for 34 different verbs. On average, target verbs have 20.5 substitution candidates; for individual instances of a target verb, an average of 3.9 of the substitution candidates are annotated as correct paraphrases. Below, we will refer to this dataset as "LST/SO."

Experimental procedure. To compute the vector space, we consider only a subset of the complete set of dependency triples extracted from the parsed Gigaword corpus. We experimented with various strategies, and found that models which consider all dependency triples exceeding certain *pmi*- and frequency thresholds perform best.

Since the dataset is rather small, we use a four-fold cross-validation method for parameter tuning: We divide the dataset into four subsets, test various parameter settings on one subset and use the parameters that perform best (in terms of GAP) to evaluate the model on the three other subsets. We consider the following parameters: *pmi*-thresholds for the dependency triples used in the computation of the first- and second-order vectors, and frequency thresholds. The parameters differ only slightly between the four subsets, and the general tendency is that good results are obtained if a low *pmi*-threshold (≤ 2) is applied to filter dependency triples used in the computation of the second-order vectors, and a relatively high *pmi*-threshold (≥ 4) to filter dependency triples in the computation of the first-order vectors. Good performing frequency thresholds are 10 or 15. The threshold values for context vectors are slightly different: a medium *pmi*-threshold between 2 and 4 and a low frequency threshold of 3.

To rank paraphrases in context, we compute contextualized vectors for the verb in the input sentence, i.e., a second order vector for the verb that is contextually constrained by the first order vectors of all its arguments, and compare them to the unconstrained (second-order) vectors of each paraphrase candidate, using cosine similarity.[3] For the first sentence in Table 1, for example, we compute $[\![\text{acquire}_{\text{SUBJ:student,OBJ:knowledge}}]\!]$ and compare it to $[\![\text{gain}]\!]$, $[\![\text{amass}]\!]$, $[\![\text{buy}]\!]$, $[\![\text{purchase}]\!]$ and so on.

Baselines. We evaluate our model against a random baseline and two variants of our model: One variant ("2^{nd} order uncontexualized") simply uses contextually unconstrained second-order vectors to rank paraphrase candidates. Comparing the full model to this variant will show how effective our method of contextualizing vectors is. The second variant ("1^{st} order contextualized") represents verbs in context by their first order vectors that specify how often the verb co-occurs with its arguments in the parsed Gigaword corpus. We compare our model to this baseline to demonstrate the benefit of (contextualized) second-order vectors. As for the full model, we use *pmi* values rather than raw frequency counts as co-occurrence statistics.

Results. For the LST/SO dataset, the generalized average precision, averaged over all instances in the dataset, is 45.94%, and the average P_{10} is 73.11%.

Table 2 compares our model to the random baseline, the two variants of our model, and previous work. As can be seen, our model improves about 8% in terms of GAP and almost 7% in terms of P_{10} upon the two variants of our model, which in turn perform 10% above the random baseline. We conclude that both the use of second-order vectors, as well as the method used to contextualize them, are very effective for the task under consideration.

The table also compares our model to the model of TDP and two different instantiations of E&P's model. The results for these three models are cited from Thater et al. (2009). We can observe that our model improves about 9% in terms of GAP and about 7% in terms of P_{10} upon previous work. Note that the results for the E&P models are based

[2] Both TDP and E&P use the British National Corpus.

[3] Note that the context information is the same for both words. With our choice of pointwise multiplication for the composition operator $\times$ we have $(\vec{v}_1 \times \vec{w}) \cdot \vec{v}_2 = \vec{v}_1 \cdot (\vec{v}_2 \times \vec{w})$. Therefore the choice of which word is contextualized does not strongly influence their cosine similarity, and contextualizing both should not add any useful information. On the contrary we found that it even lowers performance. Although this could be repaired by appropriately modifying the operator $\times$, for this experiment we stick with the easier solution of only contextualizing one of the words.

Model	GAP	P_{10}
Random baseline	26.03	54.25
E&P (add, object)	29.93	66.20
E&P (min, subject & object)	32.22	64.86
TDP	36.54	63.32
1^{st} order contextualized	36.09	59.35
2^{nd} order uncontextualized	37.65	66.32
Full model	**45.94**	**73.11**

Table 2: Results of Experiment 1

on a reimplementation of E&P's original model—the P_{10}-scores reported by Erk and Padó (2009) range between 60.2 and 62.3, over a slightly lower random baseline.

According to a paired t-test the differences are statistically significant at $p < 0.01$.

Performance on the complete dataset. To find out how our model performs on less controlled datasets, we extracted all instances from the lexical substitution task dataset with a verb target, excluding only instances which could not be parsed by the Stanford parser, or in which the target was mistagged as a non-verb by the parser. The resulting dataset contains 496 instances. As for the LST/SO dataset, we ignore all gold-standard paraphrases that do not occur in the parsed (Gigaword) corpus.

If we use the best-performing parameters from the first experiment, we obtain a GAP score of 45.17% and a P_{10}-score of 75.43%, compared to random baselines of 27.42% (GAP) and 58.83% (P_{10}). The performance on this larger dataset is thus almost the same compared to our results for the more controlled dataset. We take this as evidence that our model is quite robust w.r.t. different realizations of a verb's subcategorization frame.

4.4 Experiment 2: Non-verb paraphrases

We now apply our model to parts of speech (POS) other than verbs. The main difference between verbs on the one hand, and nouns, adjectives, and adverbs on the other hand, is that verbs typically come with a rich context—subject, object, and so on—while non-verbs often have either no dependents at all or only closed class dependents such as determiners which provide only limited contextual informations, if any at all. While we can apply the same method as before also to non-verbs, we might expect it to work less well due to limited contextual information.

POS	Instances	M1	M2	Baseline
Noun	535	46.38	42.54	30.01
Adj	508	39.41	43.21	28.32
Adv	284	48.19	51.43	37.25

Table 3: GAP-scores for non-verb paraphrases using two different methods.

We therefore propose an alternative method to rank non-verb paraphrases: We take the second-order vector of the target's head and contextually constrain it by the first order vector of the target. For instance, if we want to rank the paraphrase candidates *hint* and *star* for the noun *lead* in the sentence

(1) Meet for coffee early, swap *leads* and get permission to contact if possible.

we compute $[\![\text{swap}_{\text{OBJ:lead}}]\!]$ and compare it to the lifted first-order vectors of all paraphrase candidates, $L_{\text{OBJ}}([\text{hint}])$ and $L_{\text{OBJ}}([\text{star}])$, using cosine similarity.

To evaluate the performance of the two methods, we extract all instances from the lexical substitution task dataset with a nominal, adjectival, or adverbial target, excluding instances with incorrect parse or no parse at all. As before, we ignore gold-standard paraphrases that do not occur in the parsed Gigaword corpus.

The results are shown in Table 3, where "M1" refers to the method we used before on verbs, and "M2" refers to the alternative method described above. As one can see, M1 achieves better results than M2 if applied to nouns, while M2 is better than M1 if applied to adjectives and adverbs. The second result is unsurprising, as adjectives and adverbs often have no dependents at all.

We can observe that the performance of our model is similarly strong on non-verbs. GAP scores on nouns (using M1) and adverbs are even higher than those on verbs. We take these results to show that our model can be successfully applied to all open word classes.

5 Experiment: Ranking Word Senses

In this section, we apply our model to a different word sense ranking task: Given a word w in context, the task is to decide to what extent the different

WordNet (Fellbaum, 1998) senses of *w* apply to this occurrence of *w*.

Dataset. We use the dataset provided by Erk and McCarthy (2009). The dataset contains ordinal judgments of the applicability of WordNet senses on a 5 point scale, ranging from *completely different* to *identical* for eight different lemmas in 50 different sentential contexts. In this experiment, we concentrate on the three verbs in the dataset: *ask*, *add* and *win*.

Experimental procedure. Similar to Pennacchiotti et al. (2008), we represent different word senses by the words in the corresponding synsets. For each word sense, we compute the centroid of the second-order vectors of its synset members. Since synsets tend to be small (they even may contain only the target word itself), we additionally add the centroid of the sense's hypernyms, scaled down by the factor 10 (chosen as a rough heuristic without any attempt at optimization).

We apply the same method as in Section 4.3: For each instance in the dataset, we compute the second-order vector of the target verb, contextually constrain it by the first-order vectors of the verb's arguments, and compare the resulting vector to the vectors that represent the different WordNet senses of the verb. The WordNet senses are then ranked according to the cosine similarity between their sense vector and the contextually constrained target verb vector.

To compare the predicted ranking to the gold-standard ranking, we use Spearman's ρ, a standard method to compare ranked lists to each other. We compute ρ between the similarity scores averaged over all three annotators and our model's predictions. Based on agreement between human judges, Erk and McCarthy (2009) estimate an upper bound ρ of 0.544 for the dataset.

Results. Table 4 shows the results of our experiment. The first column shows the correlation of our model's predictions with the human judgments from the gold-standard, averaged over all instances. All correlations are significant ($p < 0.001$) as tested by approximate randomization (Noreen, 1989).

The second column shows the results of a frequency-informed baseline, which predicts the ranking based on the order of the senses in WordNet. This (weakly supervised) baseline outperforms our unsupervised model for two of the three verbs. As a final step, we explored the effect of combining our rankings with those of the frequency baseline, by simply computing the average ranks of those two models. The results are shown in the third column. Performance is significantly higher than for both the original model and the frequency-informed baseline. This shows that our model captures an additional kind of information, and thus can be used to improve the frequency-based model.

Word	Present paper	WN-Freq	Combined
ask	0.344	0.369	0.431
add	0.256	0.164	0.270
win	0.236	0.343	0.381
average	*0.279*	*0.291*	*0.361*

Table 4: Correlation of model predictions and human judgments

6 Conclusion

We have presented a novel method for adapting the vector representations of words according to their context. In contrast to earlier approaches, our model incorporates detailed syntactic information. We solved the problems of data sparseness and incompatibility of dimensions which are inherent in this approach by modeling contextualization as an interplay between first- and second-order vectors.

Evaluating on the SemEval 2007 lexical substitution task dataset, our model performs substantially better than all earlier approaches, exceeding the state of the art by around 9% in terms of generalized average precision and around 7% in terms of precision out of ten. Also, our system is the first unsupervised method that has been applied to Erk and McCarthy's (2009) graded word sense assignment task, showing a substantial positive correlation with the gold standard. We further showed that a weakly supervised heuristic, making use of WordNet sense ranks, can be significantly improved by incorporating information from our system.

We studied the effect that context has on target words in a series of experiments, which vary the target word and keep the context constant. A natural objective for further research is the influence of varying contexts on the meaning of target expressions. This extension might also shed light on the status of the modelled semantic process, which we have been referring to in this paper as "contextualization". This process can be considered one of

mutual disambiguation, which is basically the view of E&P. Alternatively, one can conceptualize it as semantic composition: in particular, the head of a phrase incorporates semantic information from its dependents, and the final result may to some extent reflect the meaning of the whole phrase.

Another direction for further study will be the generalization of our model to larger syntactic contexts, including more than only the direct neighbors in the dependency graph, ultimately incorporating context information from the whole sentence in a recursive fashion.

Acknowledgments. We would like to thank Eduard Hovy and Georgiana Dinu for inspiring discussions and helpful comments. This work was supported by the Cluster of Excellence "Multimodal Computing and Interaction", funded by the German Excellence Initiative, and the project SALSA, funded by DFG (German Science Foundation).

References

Chris Buckley and Ellen M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece.

Kenneth W. Church and Patrick Hanks. 1990. Word association, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006)*, pages 449–454, Genoa, Italy.

Dmitriy Dligach and Martha Palmer. 2008. Novel semantic features for verb sense disambiguation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 29–32, Columbus, OH, USA.

Katrin Erk and Diana McCarthy. 2009. Graded word sense assignment. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 440–449, Singapore.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, HI, USA.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proc. of the Workshop on Geometrical Models of Natural Language Semantics*, Athens, Greece.

Christiane Fellbaum, editor. 1998. *Wordnet: An Electronic Lexical Database*. Bradford Book.

Walter Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

Kazuaki Kishida. 2005. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. *NII Technical Report*.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Dekang Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 112–120, Columbus, OH, USA.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proc. of SemEval*, Prague, Czech Republic.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, OH, USA.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language*, pages 221–242. Dordrecht.

Eric W. Noreen. 1989. *Computer-intensive Methods for Testing Hypotheses: An Introduction*. John Wiley and Sons Inc.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Marco Pennacchiotti, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of framenet lexical units. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 457–465, Honolulu, HI, USA.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. 2009. Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 44–47, Singapore.

Bootstrapping Semantic Analyzers from Non-Contradictory Texts

Ivan Titov **Mikhail Kozhevnikov**
Saarland University
Saarbrücken, Germany
{titov|m.kozhevnikov}@mmci.uni-saarland.de

Abstract

We argue that groups of unannotated texts with overlapping and non-contradictory semantics represent a valuable source of information for learning semantic representations. A simple and efficient inference method recursively induces joint semantic representations for each group and discovers correspondence between lexical entries and latent semantic concepts. We consider the generative semantics-text correspondence model (Liang et al., 2009) and demonstrate that exploiting the non-contradiction relation between texts leads to substantial improvements over natural baselines on a problem of analyzing human-written weather forecasts.

1 Introduction

In recent years, there has been increasing interest in statistical approaches to semantic parsing. However, most of this research has focused on supervised methods requiring large amounts of labeled data. The supervision was either given in the form of meaning representations aligned with sentences (Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Mooney, 2007) or in a somewhat more relaxed form, such as lists of candidate meanings for each sentence (Kate and Mooney, 2007; Chen and Mooney, 2008) or formal representations of the described world state for each text (Liang et al., 2009). Such annotated resources are scarce and expensive to create, motivating the need for unsupervised or semi-supervised techniques (Poon and Domingos, 2009). However, unsupervised methods have their own challenges: they are not always able to discover semantic equivalences of lexical entries or logical forms or, on the contrary, cluster semantically different or even opposite expressions (Poon and Domingos, 2009). Unsupervised approaches can only rely on distributional similarity of contexts (Harris, 1968) to decide on semantic relatedness of terms, but this information may be sparse and not reliable (Weeds and Weir, 2005). For example, when analyzing weather forecasts it is very hard to discover in an unsupervised way which of the expressions among "south wind", "wind from west" and "southerly" denote the same wind direction and which are not, as they all have a very similar distribution of their contexts. The same challenges affect the problem of identification of argument roles and predicates.

In this paper, we show that groups of unannotated texts with overlapping and non-contradictory semantics provide a valuable source of information. This form of weak supervision helps to discover implicit clustering of lexical entries and predicates, which presents a challenge for purely unsupervised techniques. We assume that each text in a group is independently generated from a full latent semantic state corresponding to the group. Importantly, the texts in each group do not have to be paraphrases of each other, as they can verbalize only specific parts (*aspects*) of the full semantic state, yet statements about the same aspects must not contradict each other. Simultaneous inference of the semantic state for the non-contradictory and semantically overlapping documents would restrict the space of compatible hypotheses, and, intuitively, 'easier' texts in a group will help to analyze the 'harder' ones.[1]

As an illustration of why this weak supervision may be valuable, consider a group of two non-contradictory texts, where one text mentions "2.2 bn GBP decrease in profit", whereas another one includes a passage "profit fell by 2.2 billion pounds". Even if the model has not observed

[1] This view on this form of supervision is evocative of co-training (Blum and Mitchell, 1998) which, roughly, exploits the fact that the same example can be 'easy' for one model but 'hard' for another one.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 958–967,
Uppsala, Sweden, 11-16 July 2010.

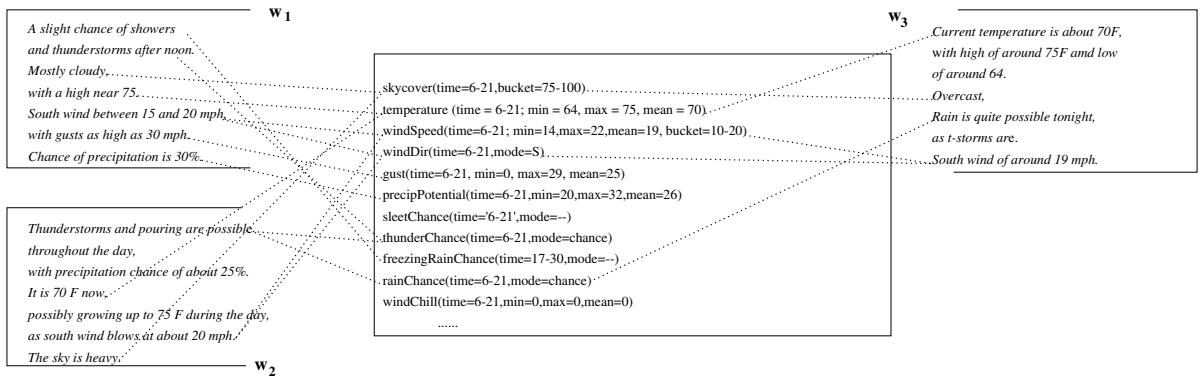


Figure 1: An example of three non-contradictory weather forecasts and their alignment to the semantic representation. Note that the semantic representation (the block in the middle) is not observable in training.

the word "fell" before, it is likely to align these phrases to the same semantic form because of similarity of their arguments. And this alignment would suggest that "fell" and "decrease" refer to the same process, and should be clustered together. This would not happen for the pair "fell" and "increase" as similarity of their arguments would normally entail contradiction. Similarly, in the example mentioned earlier, when describing a forecast for a day with expected south winds, texts in the group can use either "south wind" or "southerly" to indicate this fact but no texts would verbalize it as "wind from west", and therefore these expressions will be assigned to different semantic clusters. However, it is important to note that the phrase "wind from west" may still appear in the texts, but in reference to other time periods, underlying the need for modeling alignment between grouped texts and their latent meaning representation.

As much of the human knowledge is redescribed multiple times, we believe that non-contradictory and semantically overlapping texts are often easy to obtain. For example, consider semantic analysis of news articles or biographies. In both cases we can find groups of documents referring to the same events or persons, and though they will probably focus on different aspects and have different subjective passages, they are likely to agree on the core information (Shinyama and Sekine, 2003). Alternatively, if such groupings are not available, it may still be easier to give each semantic representation (or a state) to multiple annotators and ask each of them to provide a textual description, instead of annotating texts with semantic expressions. The state can be communicated to them in a visual or audio form (e.g., as a picture or a short video clip) ensuring that their interpretations are consistent.

Unsupervised learning with shared latent semantic representations presents its own challenges, as exact inference requires marginalization over possible assignments of the latent semantic state, consequently, introducing non-local statistical dependencies between the decisions about the semantic structure of each text. We propose a simple and fairly general approximate inference algorithm for probabilistic models of semantics which is efficient for the considered model, and achieves favorable results in our experiments.

In this paper, we do not consider models which aim to produce complete formal meaning of text (Zettlemoyer and Collins, 2005; Mooney, 2007; Poon and Domingos, 2009), instead focusing on a simpler problem studied in (Liang et al., 2009). They investigate grounded language acquisition set-up and assume that semantics (*world state*) can be represented as a set of *records* each consisting of a set of *fields*. Their model segments text into utterances and identifies records, fields and field values discussed in each utterance. Therefore, one can think of this problem as an extension of the semantic role labeling problem (Carreras and Marquez, 2005), where predicates (i.e. *records* in our notation) and their arguments should be identified in text, but here arguments are not only assigned to a specific role (*field*) but also mapped to an underlying equivalence class (*field value*). For example, in the weather forecast domain field *sky cover* should get the same value given expressions "overcast" and "very cloudy" but a different one if the expres-

sions are "clear" or "sunny". This model is hard to evaluate directly as text does not provide information about all the fields and does not necessarily provide it at the sufficient granularity level. Therefore, it is natural to evaluate their model on the database-text alignment problem (Snyder and Barzilay, 2007), i.e. measuring how well the model predicts the alignment between the text and the observable records describing the entire world state. We follow their set-up, but assume that instead of having access to the full semantic state for every training example, we have a very small amount of data annotated with semantic states and a larger number of unannotated texts with non-contradictory semantics.

We study our set-up on the weather forecast data (Liang et al., 2009) where the original textual weather forecasts were complemented by additional forecasts describing the same weather states (see figure 1 for an example). The average overlap between the verbalized fields in each group of non-contradictory forecasts was below 35%, and more than 60% of fields are mentioned only in a single forecast from a group. Our model, learned from 100 labeled forecasts and 259 groups of unannotated non-contradictory forecasts (750 texts in total), achieved 73.9% F_1. This compares favorably with 69.1% shown by a semi-supervised learning approach, though, as expected, does not reach the score of the model which, in training, observed semantics states for all the 750 documents (77.7% F_1).

The rest of the paper is structured as follows. In section 2 we describe our inference algorithm for groups of non-contradictory documents. Section 3 redescribes the semantics-text correspondence model (Liang et al., 2009) in the context of our learning scenario. In section 4 we provide an empirical evaluation of the proposed method. We conclude in section 5 with an examination of additional related work.

2 Inference with Non-Contradictory Documents

In this section we will describe our inference method on a higher conceptual level, not specifying the underlying meaning representation and the probabilistic model. An instantiation of the algorithm for the semantics-text correspondence model is given in section 3.2.

Statistical models of parsing can often be regarded as defining the probability distribution of meaning $\boldsymbol{m}$ and its alignment $\boldsymbol{a}$ with the given text $\boldsymbol{w}$, $P(\boldsymbol{m}, \boldsymbol{a}, \boldsymbol{w}) = P(\boldsymbol{a}, \boldsymbol{w}|\boldsymbol{m})P(\boldsymbol{m})$. The semantics $\boldsymbol{m}$ can be represented either as a logical formula (see, e.g., (Poon and Domingos, 2009)) or as a set of field values if database records are used as a meaning representation (Liang et al., 2009). The alignment $\boldsymbol{a}$ defines how semantics is verbalized in the text $\boldsymbol{w}$, and it can be represented by a meaning derivation tree in case of full semantic parsing (Poon and Domingos, 2009) or, e.g., by a hierarchical segmentation into utterances along with an utterance-field alignment in a more shallow variation of the problem. In semantic parsing, we aim to find the most likely underlying semantics and alignment given the text:

$$(\hat{\boldsymbol{m}}, \hat{\boldsymbol{a}}) = \arg\max_{\boldsymbol{m},\boldsymbol{a}} P(\boldsymbol{a}, \boldsymbol{w}|\boldsymbol{m})P(\boldsymbol{m}). \quad (1)$$

In the supervised case, where $\boldsymbol{a}$ and $\boldsymbol{m}$ are observable, estimation of the generative model parameters is generally straightforward. However, in a semi-supervised or unsupervised case variational techniques, such as the EM algorithm (Dempster et al., 1977), are often used to estimate the model. As common for complex generative models, the most challenging part is the computation of the posterior distributions $P(\boldsymbol{a}, \boldsymbol{m}|\boldsymbol{w})$ on the E-step which, depending on the underlying model $P(\boldsymbol{m}, \boldsymbol{a}, \boldsymbol{w})$, may require approximate inference.

As discussed in the introduction, our goal is to integrate groups of non-contradictory documents into the learning procedure. Let us denote by $\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K$ a group of non-contradictory documents. As before, the estimation of the posterior probabilities $P(\boldsymbol{m}_i, \boldsymbol{a}_i|\boldsymbol{w}_1 \ldots \boldsymbol{w}_K)$ presents the main challenge. Note that the decision about $\boldsymbol{m}_i$ is now conditioned on all the texts $\boldsymbol{w}_j$ rather than only on $\boldsymbol{w}_i$. This conditioning is exactly what drives learning, as the information about likely semantics $\boldsymbol{m}_j$ of text j affects the decision about choice of $\boldsymbol{m}_i$:

$$P(\boldsymbol{m}_i|\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) \propto \sum_{a_i} P(\boldsymbol{a}_i, \boldsymbol{w}_i|\boldsymbol{m}_i) \times$$
$$\times \sum_{\boldsymbol{m}_{-i},\boldsymbol{a}_{-i}} P(\boldsymbol{m}_i|\boldsymbol{m}_{-i})P(\boldsymbol{m}_{-i}, \boldsymbol{a}_{-i}, \boldsymbol{w}_{-i}), \quad (2)$$

where $\boldsymbol{x}_{-i}$ denotes $\{\boldsymbol{x}_j : j \neq i\}$. $P(\boldsymbol{m}_i|\boldsymbol{m}_{-i})$ is the probability of the semantics $\boldsymbol{m}_i$ given all the meanings $\boldsymbol{m}_{-i}$. This probability assigns zero weight to inconsistent meanings, i.e. such mean-

ings $(\boldsymbol{m_1},...,\boldsymbol{m_K})$ that $\wedge_{i=1}^{K}\boldsymbol{m}_i$ is not satisfiable,[2] and models dependencies between components in the composite meaning representation (e.g., arguments values of predicates). As an illustration, in the forecast domain it may express that clouds, and not sunshine, are likely when it is raining. Note, that this probability is different from the probability that $\boldsymbol{m}_i$ is actually verbalized in the text.

Unfortunately, these dependencies between $\boldsymbol{m}_i$ and $\boldsymbol{w}_j$ are non-local. Even though the dependencies are only conveyed via $\{\boldsymbol{m}_j : j \neq i\}$ the space of possible meanings $\boldsymbol{m}$ is very large even for relatively simple semantic representations, and, therefore, we need to resort to efficient approximations.

One natural approach would be to use a form of belief propagation (Pearl, 1982; Murphy et al., 1999), where messages pass information about likely semantics between the texts. However, this approach is still expensive even for simple models, both because of the need to represent distributions over $\boldsymbol{m}$ and also because of the large number of iterations of message exchange needed to reach convergence (if it converges).

An even simpler technique would be to parse texts in a random order conditioning each meaning $\boldsymbol{m}_k^\star$ for $k \in \{1,...,K\}$ on all the previous semantics $\boldsymbol{m}_{<k}^\star = \boldsymbol{m}_1^\star,...,\boldsymbol{m}_{k-1}^\star$:

$$\boldsymbol{m}_k^\star = \arg\max_{\boldsymbol{m}_k} P(\boldsymbol{w}_k|\boldsymbol{m}_k)P(\boldsymbol{m}_k|\boldsymbol{m}_{<k}^\star).$$

Here, and in further discussion, we assume that the above search problem can be efficiently solved, exactly or approximately. However, a major weakness of this algorithm is that decisions about components of the composite semantic representation (e.g., argument values) are made only on the basis of a single text, which first mentions the corresponding aspects, without consulting any future texts $k' > k$, and these decisions cannot be revised later.

We propose a simple algorithm which aims to find an appropriate order of the greedy inference by estimating how well each candidate semantics $\hat{\boldsymbol{m}}_k$ would explain other texts and at each step selecting k (and $\hat{\boldsymbol{m}}_k$) which explains them best.

The algorithm, presented in figure 2[3], constructs an ordering of texts $\boldsymbol{n} = (n_1,...,n_K)$

1: $\boldsymbol{n} := (), \boldsymbol{m}^\star := ()$
2: **for** $i := 1 : K-1$ **do**
3: **for** $j \notin \boldsymbol{n}$ **do**
4: $\hat{\boldsymbol{m}}_j := \arg\max_{\boldsymbol{m}_j} P(\boldsymbol{m}_j, \boldsymbol{w}_j|\boldsymbol{m}^\star)$
5: **end for**
6: $n_i := \arg\max_{j\notin\boldsymbol{n}} P(\hat{\boldsymbol{m}}_j, \boldsymbol{w}_j|\boldsymbol{m}^\star)\times$
$\times \prod_{k\notin\boldsymbol{n}\cup\{j\}} \max_{\boldsymbol{m}_k} P(\boldsymbol{m}_k, \boldsymbol{w}_k|\boldsymbol{m}^\star, \hat{\boldsymbol{m}}_j)$
7: $\boldsymbol{m}_i^\star := \hat{\boldsymbol{m}}_{n_i}$
8: **end for**
9: $n_K := \{1,...,K\}\backslash\boldsymbol{n}$
10: $\boldsymbol{m}_K^\star := \arg\max_{\boldsymbol{m}_{n_K}} P(\boldsymbol{m}_{n_K}, \boldsymbol{w}_{n_K}|\boldsymbol{m}^\star)$

Figure 2: The approximate inference algorithm.

and corresponding meaning representations $\boldsymbol{m}^\star = (\boldsymbol{m}_1^\star,...,\boldsymbol{m}_K^\star)$, where $\boldsymbol{m}_k^\star$ is the predicted meaning representation of text $\boldsymbol{w}_{n_k}$. It starts with an empty ordering $\boldsymbol{n} = ()$ and an empty list of meanings $\boldsymbol{m}^\star = ()$ (line 1). Then it iteratively predicts meaning representations $\hat{\boldsymbol{m}}_j$ conditioned on the list of semantics $\boldsymbol{m}^\star = (\boldsymbol{m}_1^\star,...,\boldsymbol{m}_{i-1}^\star)$ fixed on the previous stages and does it for all the remaining texts $\boldsymbol{w}_j$ (lines 3-5). The algorithm selects a single meaning $\hat{\boldsymbol{m}}_j$ which maximizes the probability of all the remaining texts and excludes the text j from future consideration (lines 6-7).

Though the semantics $\boldsymbol{m}_k$ ($k \notin \boldsymbol{n}\cup\{j\}$) used in the estimates (line 6) can be inconsistent with each other, the final list of meanings $\boldsymbol{m}^\star$ is guaranteed to be consistent. It holds because on each iteration we add a single meaning $\hat{\boldsymbol{m}}_{n_i}$ to $\boldsymbol{m}^\star$ (line 7), and $\hat{\boldsymbol{m}}_{n_i}$ is guaranteed to be consistent with $\boldsymbol{m}^\star$, as the semantics $\hat{\boldsymbol{m}}_{n_i}$ was conditioned on the meaning $\boldsymbol{m}^\star$ during inference (line 4).

An important aspect of this algorithm is that unlike usual greedy inference, the remaining ('future') texts do affect the choice of meaning representations made on the earlier stages. As soon as semantics $\boldsymbol{m}_k^\star$ are inferred for every k, we find ourselves in the set-up of learning with unaligned semantic states considered in (Liang et al., 2009).

The induced alignments $\boldsymbol{a}_1,...,\boldsymbol{a}_K$ of semantics $\boldsymbol{m}^\star$ to texts $\boldsymbol{w}_1,...,\boldsymbol{w}_K$ at the same time induce alignments between the texts. The problem of producing multiple sequence alignment, especially in the context of sentence alignments, has been extensively studied in NLP (Barzilay and Lee, 2003). In this paper, we use semantic structures as a pivot for finding the best alignment in the hope that presence of meaningful text alignments will improve the quality of the resulting semantic structures by enforcing a form of agreement between them.

[2]Note that checking for satisfiability may be expensive or intractable depending on the formalism.

[3]We slightly abuse notation by using set operations with the lists $\boldsymbol{n}$ and $\boldsymbol{m}^\star$ as arguments. Also, for all the document indices j we use $j \notin S$ to denote $j \in \{1,...,K\}\backslash S$.

3 A Model of Semantics

In this section we redescribe the semantics-text correspondence model (Liang et al., 2009) with an extension needed to model examples with latent states, and also explain how the inference algorithm defined in section 2 can be applied to this model.

3.1 Model definition

Liang et al. (2009) considered a scenario where each text was annotated with a world state, even though alignment between the text and the state was not observable. This is a weaker form of supervision than the one traditionally considered in supervised semantic parsing, where the alignment is also usually provided in training (Chen and Mooney, 2008; Zettlemoyer and Collins, 2005). Nevertheless, both in training and testing the world state is observable, and the alignment and the text are conditioned on the state during inference. Consequently, there was no need to model the distribution of the world state. This is different for us, and we augment the generative story by adding a simplistic world state generation step.

As explained in the introduction, the world states $\boldsymbol{s}$ are represented by sets of records (see the block in the middle of figure 1 for an example of a world state). Each record is characterized by a record type $t \in \{1,...,T\}$, which defines the set of fields $\boldsymbol{F}^{(t)}$. There are $n^{(t)}$ records of type t and this number may change from document to document. For example, there may be more than a single record of type *wind speed*, as they may refer to different time periods but all these records have the same set of fields, such as minimal, maximal and average wind speeds. Each field has an associated type: in our experiments we consider only categorical and integer fields. We write $s^{(t)}_{n,f} = v$ to denote that n-th record of type t has field f set to value v.

Each document k verbalizes a subset of the entire world state, and therefore semantics $\boldsymbol{m}_k$ of the document is an assignment to $|\boldsymbol{m}_k|$ verbalized fields: $\wedge_{q=1}^{|\boldsymbol{m}_k|}(s^{(t_q)}_{n_q,f_q} = v_q)$, where t_q, n_q, f_q are the verbalized record types, records and fields, respectively, and v_q is the assigned field value. The probability of meaning m_k then equals the probability of this assignment with other state variables left non-observable (and therefore marginalized out). In this formalism checking for contradiction is trivial: two meaning representations contradict each other if they assign different values to the same field of the same record.

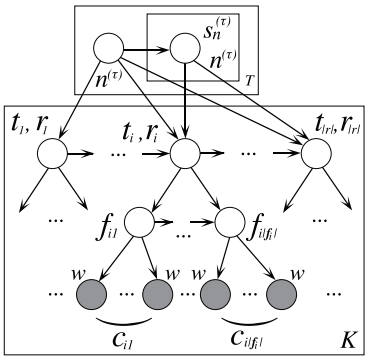


Figure 3: The semantics-text correspondence model with K documents sharing the same latent semantic state.

The semantics-text correspondence model defines a hierarchical segmentation of text: first, it segments the text into fragments discussing different records, then the utterances corresponding to each record are further segmented into fragments verbalizing specific fields of that record. An example of a segmented fragment is presented in figure 4. The model has a designated null-record which is aligned to words not assigned to any record. Additionally there is a null-field in each record to handle words not specific to any field. In figure 3 the corresponding graphical model is presented. The formal definition of the model for documents $\boldsymbol{w_1},..., \boldsymbol{w_K}$ sharing a semantic state is as follows:

- Generation of world state $\boldsymbol{s}$:
  - For each type $\tau \in \{1,...,T\}$ choose a number of records of that type $n^{(\tau)} \sim \text{Unif}(1,..., n_{max})$.
  - For each record $\boldsymbol{s}^{(\tau)}_n$, $n \in \{1,..,n^{(\tau)}\}$ choose field values $s^{(\tau)}_{nf}$ for all fields $f \in \boldsymbol{F}^{(\tau)}$ from the type-specific distribution.
- Generation of the verbalizations, for each document $\boldsymbol{w}_k$, $k \in \{1,..., K\}$:[4]
  - Record Types: Choose a sequence of verbalized record types $\boldsymbol{t} = (t_1,..., t_{|\boldsymbol{t}|})$ from the first-order Markov chain.
  - Records: For each type t_i choose a verbalized record $\boldsymbol{r}_i$ from all the records of that type: $l \sim \text{Unif}(1,..., n^{(\tau)})$, $\boldsymbol{r}_i := \boldsymbol{s}^{(t_i)}_l$.
  - Fields: For each record $\boldsymbol{r}_i$ choose a sequence of verbalized fields $\boldsymbol{f}_i = (f_{i1},..., f_{i|\boldsymbol{f}_i|})$ from the first-order Markov chain ($f_{ij} \in \boldsymbol{F}^{(t_i)}$).
  - Length: For each field f_{ij}, choose length $c_{ij} \sim \text{Unif}(1,..., c_{max})$.
  - Words: Independently generate c_{ij} words from the field-specific distribution $P(w|f_{ij}, r_{if_{ij}})$.

[4] We omit index k in the generative story and figure 3 to simplify the notation.

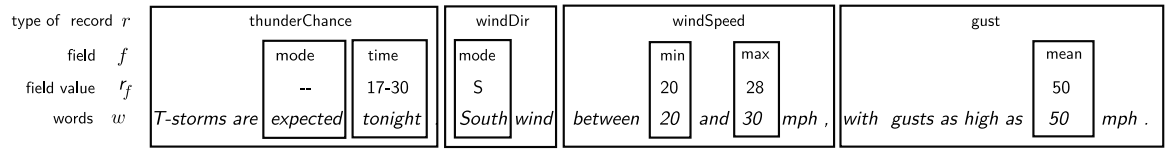


Figure 4: A segmentation of a text fragment into records and fields.

Note that, when generating fields, the Markov chain is defined over fields and the transition parameters are independent of the field values $r_{if_{ij}}$. On the contrary, when drawing a word, the distribution of words is conditioned on the value of the corresponding field.

The form of word generation distributions $P(w|f_{ij}, r_{if_{ij}})$ depends on the type of the field $f_{i,j}$. For categorical fields, the distribution of words is modeled as a distinct multinomial for each field value. Verbalizations of numerical fields are generated via a perturbation on the field value $r_{if_{ij}}$: the value $r_{if_{ij}}$ can be perturbed by either rounding it (up or down) or distorting (up or down, modeled by a geometric distribution). The parameters corresponding to each form of generation are estimated during learning. For details on these emission models, as well as for details on modeling record and field transitions, we refer the reader to the original publication (Liang et al., 2009).

In our experiments, when choosing a world state $\boldsymbol{s}$, we generate the field values independently. This is clearly a suboptimal regime as often there are very strong dependencies between field values: e.g., in the weather domain many record types contain groups of related fields defining minimal, maximal and average values of some parameter. Extending the method to model, e.g., pairwise dependencies between field values is relatively straightforward.

As explained above, semantics of a text $\boldsymbol{m}$ is defined by the assignment of state variables $\boldsymbol{s}$. Analogously, an alignment $\boldsymbol{a}$ between semantics $\boldsymbol{m}$ and a text $\boldsymbol{w}$ is represented by all the remaining latent variables: by the sequence of record types $\boldsymbol{t} = (t_1,...,t_{|\boldsymbol{t}|})$, choice of records $\boldsymbol{r}_i$ for each t_i, the field sequence $\boldsymbol{f}_i$ and the segment length c_{ij} for every field f_{ij}.

3.2 Learning and inference

We select the model parameters θ by maximizing the marginal likelihood of the data, where the data $\mathcal{D}$ is given in the form of groups $\boldsymbol{w} = \{\boldsymbol{w}_1,...,\boldsymbol{w}_K\}$ sharing the same latent state:[5]

$$\max_{\theta} \prod_{\boldsymbol{w}\in\mathcal{D}} \sum_{\boldsymbol{s}} P(\boldsymbol{s}) \prod_{k} \sum_{\boldsymbol{r},\boldsymbol{f},\boldsymbol{c}} P(\boldsymbol{r}, \boldsymbol{f}, \boldsymbol{c}, \boldsymbol{w}_k|\boldsymbol{s}, \theta).$$

To estimate the parameters, we use the Expectation-Maximization algorithm (Dempster et al., 1977). When the world state is observable, learning does not require any approximations, as dynamic programming (a form of the forward-backward algorithm) can be used to infer the posterior distribution on the E-step (Liang et al., 2009). However, when the state is latent, dependencies are not local anymore, and approximate inference is required.

We use the algorithm described in section 2 (figure 2) to infer the state. In the context of the semantics-text correspondence model, as we discussed above, semantics $\boldsymbol{m}$ defines the subset of admissible world states. In order to use the algorithm, we need to understand how the conditional probabilities of the form $P(\boldsymbol{m}'|\boldsymbol{m})$ are computed, as they play the key role in the inference procedure (see equation (2)). If there is a contradiction ($\boldsymbol{m}' \perp \boldsymbol{m}$) then $P(\boldsymbol{m}'|\boldsymbol{m}) = 0$, conversely, if $\boldsymbol{m}'$ is subsumed by $\boldsymbol{m}$ ($\boldsymbol{m} \rightarrow \boldsymbol{m}'$) then this probability is 1. Otherwise, $P(\boldsymbol{m}'|\boldsymbol{m})$ equals the probability of new assignments $\wedge_{q=1}^{|\boldsymbol{m}'\setminus\boldsymbol{m}|}(s_{n'_q,f'_q}^{(t'_q)} = v'_q)$ (defined by $\boldsymbol{m}'\setminus\boldsymbol{m}$) conditioned on the previously fixed values of $\boldsymbol{s}$ (given by $\boldsymbol{m}$). Summarizing, when predicting the most likely semantics $\hat{\boldsymbol{m}}_j$ (line 4), for each span the decoder weighs alternatives of either (1) aligning this span to the previously induced meaning $\boldsymbol{m}^\star$, or (2) aligning it to a new field and paying the cost of generation of its value.

The exact computation of the most probable semantics (line 4 of the algorithm) is intractable, and we have to resort to an approximation. Instead of predicting the most probable semantics $\hat{\boldsymbol{m}}_j$ we search for the most probable pair $(\hat{\boldsymbol{a}}_j, \hat{\boldsymbol{m}}_j)$, thus assuming that the probability mass is mostly concentrated on a single alignment. The alignment $\boldsymbol{a}_j$

[5]For simplicity, we assume here that all the examples are unlabeled.

is then discarded and not used in any other computations. Though the most likely alignment $\hat{\boldsymbol{a}}_j$ for a fixed semantic representation $\hat{\boldsymbol{m}}_j$ can be found efficiently using a Viterbi algorithm, computing the most probable pair $(\hat{\boldsymbol{a}}_j, \hat{\boldsymbol{m}}_j)$ is still intractable. We use a modification of the beam search algorithm, where we keep a set of candidate meanings (partial semantic representations) and compute an alignment for each of them using a form of the Viterbi algorithm.

As soon as the meaning representations $\boldsymbol{m}^\star$ are inferred, we find ourselves in the set-up studied in (Liang et al., 2009): the state $\boldsymbol{s}$ is no longer latent and we can run efficient inference on the E-step. Though some fields of the state $\boldsymbol{s}$ may still not be specified by $\boldsymbol{m}^\star$, we prohibit utterances from aligning to these non-specified fields.

On the M-step of EM the parameters are estimated as proportional to the expected marginal counts computed on the E-step. We smooth the distributions of values for numerical fields with convolution smoothing equivalent to the assumption that the fields are affected by distortion in the form of a two-sided geometric distribution with the success rate parameter equal to 0.67. We use add-0.1 smoothing for all the remaining multinomial distributions.

4 Empirical Evaluation

In this section, we consider the semi-supervised set-up, and present evaluation of our approach on on the problem of aligning weather forecast reports to the formal representation of weather.

4.1 Experiments

To perform the experiments we used a subset of the weather dataset introduced in (Liang et al., 2009). The original dataset contains 22,146 texts of 28.7 words on average, there are 12 types of records (predicates) and 36.0 records per forecast on average. We randomly chose 100 texts along with their world states to be used as the labeled data.[6] To produce groups of non-contradictory texts we have randomly selected a subset of weather states, represented them in a visual form (icons accompanied by numerical and symbolic parameters) and then manually annotated these illustrations. These newly-produced forecasts, when combined with the original texts, resulted in 259 groups of non-contradictory texts (650 texts, 2.5 texts per group). An example of such a group is given in figure 1.

The dataset is relatively noisy: there are inconsistencies due to annotation mistakes (e.g., number distortions), or due to different perception of the weather by the annotators (e.g., expressions such as 'warm' or 'cold' are subjective). The overlap between the verbalized fields in each group was estimated to be below 35%. Around 60% of fields are mentioned only in a single forecast from a group, consequently, the texts cannot be regarded as paraphrases of each other.

The test set consists of 150 texts, each corresponding to a different weather state. Note that during testing we no longer assume that documents share the state, we treat each document in isolation. We aimed to preserve approximately the same proportion of new and original examples as we had in the training set, therefore, we combined 50 texts originally present in the weather dataset with additional 100 newly-produced texts. We annotated these 100 texts by aligning each line to one or more records,[7] whereas for the original texts the alignments were already present. Following Liang et al. (2009) we evaluate the models on how well they predict these alignments.

When estimating the model parameters, we followed the training regime prescribed in (Liang et al., 2009). Namely, 5 iterations of EM with a basic model (with no segmentation or coherence modeling), followed by 5 iterations of EM with the model which generates fields independently and, at last, 5 iterations with the full model. Only then, in the semi-supervised learning scenarios, we added unlabeled data and ran 5 additional iterations of EM.

Instead of prohibiting records from crossing punctuation, as suggested by Liang et al. (2009), in our implementation we disregard the words not attached to specific fields (attached to the null-field, see section 3.1) when computing spans of records. To speed-up training, only a single record of each type is allowed to be generated when running inference for unlabeled examples on the E-

[6] In order to distinguish from completely unlabeled examples, we refer to examples labeled with world states as *labeled* examples. Note though that the alignments are not observable even for these labeled examples. Similarly, we call the models trained from this data *supervised* though full supervision was not available.

[7] The text was automatically tokenized and segmented into lines, with line breaks at punctuation characters. Information about the line breaks is not used during learning and inference.

	P	R	F_1
Supervised BL	63.3	52.9	57.6
Semi-superv BL	68.8	69.4	69.1
Semi-superv, non-contr	**78.8**	**69.5**	**73.9**
Supervised UB	69.4	88.6	77.9

Table 1: Results (precision, recall and F_1) on the weather forecast dataset.

value	top words
0-25	clear, small, cloudy, gaps, sun
25-50	clouds, increasing, heavy, produce, could
50-75	cloudy, mostly, high, cloudiness, breezy
75-100	amounts, rainfall, inch, new, possibly

Table 2: Top 5 words in the word distribution for field *mode* of record *sky cover*, function words and punctuation are omitted.

step of the EM algorithm, as it significantly reduces the search space. Similarly, though we preserved all records which refer to the first time period, for other time periods we removed all the records which declare that the corresponding event (e.g., rain or snowfall) is not expected to happen. This preprocessing results in the oracle recall of 93%.

We compare our approach (*Semi-superv, non-contr*) with two baselines: the basic supervised training on 100 labeled forecasts (*Supervised BL*) and with the semi-supervised training which disregards the non-contradiction relations (*Semi-superv BL*). The learning regime, the inference procedure and the texts for the semi-supervised baseline were identical to the ones used for our approach, the only difference is that all the documents were modeled as independent. Additionally, we report the results of the model trained with all the 750 texts labeled (*Supervised UB*), its scores can be regarded as an upper bound on the results of the semi-supervised models. The results are reported in table 1.

4.2 Discussion

Our training strategy results in a substantially more accurate model, outperforming both the supervised and semi-supervised baselines. Surprisingly, its precision is higher than that of the model trained on 750 labeled examples, though admittedly it is achieved at a very different recall level. The estimation of the model with our approach takes around one hour on a standard desktop PC, which is comparable to 40 minutes required to train the semi-supervised baseline.

In these experiments, we consider the problem of predicting alignment between text and the corresponding observable world state. The direct evaluation of the meaning recognition (i.e. semantic parsing) accuracy is not possible on this dataset, as the data does not contain information which fields are discussed. Even if it would provide this information, the documents do not verbalize the state at the necessary granularity level to predict the field values. For example, it is not possible to decide to which bucket of the field *sky cover* the expression 'cloudy' refers to, as it has a relatively uniform distribution across 3 (out of 4) buckets. The problem of predicting text-meaning alignments is interesting in itself, as the extracted alignments can be used in training of a statistical generation system or information extractors, but we also believe that evaluation on this problem is an appropriate test for the relative comparison of the semantic analyzers' performance. Additionally, note that the success of our weakly-supervised scenario indirectly suggests that the model is sufficiently accurate in predicting semantics of an unlabeled text, as otherwise there would be no useful information passed in between semantically overlapping documents during learning and, consequently, no improvement from sharing the state.[8]

To confirm that the model trained by our approach indeed assigns new words to correct fields and records, we visualize top words for the field characterizing sky cover (table 2). Note that the words "sun", "cloudiness" or "gaps" were not appearing in the labeled part of the data, but seem to be assigned to correct categories. However, correlation between rain and overcast, as also noted in (Liang et al., 2009), results in the wrong assignment of the rain-related words to the field value corresponding to very cloudy weather.

5 Related Work

Probably the most relevant prior work is an approach to bootstrapping lexical choice of a generation system using a corpus of alternative pas-

[8] We conducted preliminary experiments on synthetic data generated from a random semantic-correspondence model. Our approach outperformed the baselines both in predicting 'text'-state correspondence and in the F_1 score on the predicted set of field assignments ('text meanings').

sages (Barzilay and Lee, 2002), however, in their work all the passages were annotated with unaligned semantic expressions. Also, they assumed that the passages are paraphrases of each other, which is stronger than our non-contradiction assumption. Sentence and text alignment has also been considered in the related context of paraphrase extraction (see, e.g., (Dolan et al., 2004; Barzilay and Lee, 2003)) but this prior work did not focus on inducing or learning semantic representations. Similarly, in information extraction, there have been approaches for pattern discovery using comparable monolingual corpora (Shinyama and Sekine, 2003) but they generally focused only on discovery of a single pattern from a pair of sentences or texts.

Radev (2000) considered types of potential relations between documents, including contradiction, and studied how this information can be exploited in NLP. However, this work considered primarily multi-document summarization and question answering problems.

Another related line of research in machine learning is clustering or classification with constraints (Basu et al., 2004), where supervision is given in the form of constraints. Constraints declare which pairs of instances are required to be assigned to the same class (or required to be assigned to different classes). However, we are not aware of any previous work that generalized these methods to structured prediction problems, as trivial equality/inequality constraints are probably too restrictive, and a notion of consistency is required instead.

6 Summary and Future Work

In this work we studied the use of weak supervision in the form of non-contradictory relations between documents in learning semantic representations. We argued that this type of supervision encodes information which is hard to discover in an unsupervised way. However, exact inference for groups of documents with overlapping semantic representation is generally prohibitively expensive, as the shared latent semantics introduces nonlocal dependences between semantic representations of individual documents. To combat it, we proposed a simple iterative inference algorithm. We showed how it can be instantiated for the semantics-text correspondence model (Liang et al., 2009) and evaluated it on a dataset of weather forecasts. Our approach resulted in an improvement over the scores of both the supervised baseline and of the traditional semi-supervised learning.

There are many directions we plan on investigating in the future for the problem of learning semantics with non-contradictory relations. A promising and challenging possibility is to consider models which induce full semantic representations of meaning. Another direction would be to investigate purely unsupervised set-up, though it would make evaluation of the resulting method much more complex. One potential alternative would be to replace the initial supervision with a set of posterior constraints (Graca et al., 2008) or generalized expectation criteria (McCallum et al., 2007).

Acknowledgements

The authors acknowledge the support of the Excellence Cluster on Multimodal Computing and Interaction (MMCI). Thanks to Alexandre Klementiev, Alexander Koller, Manfred Pinkal, Dan Roth, Caroline Sporleder and the anonymous reviewers for their suggestions, and to Percy Liang for answering questions about his model.

References

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 164–171.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Sugatu Basu, Arindam Banjeree, and Raymond Mooney. 2004. Active semi-supervision for pairwise constrained clustering. In *Proc. of the SIAM International Conference on Data Mining (SDM)*, pages 333–344.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 209–214.

Xavier Carreras and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, Ann Arbor, MI USA.

David L. Chen and Raymond L. Mooney. 2008. Learning to sportcast: A test of grounded language acquisition. In *Proc. of International Conference on Machine Learning*, pages 128–135.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithms. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

P. Diaconis and B. Efron. 1983. Computer-intensive methods in statistics. *Scientific American*, pages 116–130.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the Conference on Computational Linguistics (COLING)*, pages 350–356.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL-05)*, Ann Arbor, Michigan.

Joao Graca, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. *Advances in Neural Information Processing Systems 20 (NIPS)*.

Zellig Harris. 1968. *Mathematical structures of language*. Wiley.

Rohit J. Kate and Raymond J. Mooney. 2007. Learning language semantics from ambigous supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 895–900.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of the Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Andrew McCallum, Gideon Mann, and Gregory Druck. 2007. Generalized expectation criteria. Technical Report TR 2007-60, University of Massachusetts, Amherst, MA.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 982–991.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, pages 467–475.

Judea Pearl. 1982. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 133–136.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, (EMNLP-09)*.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources step one: Cross-document structure. In *1st SIGdial Workshop on Discourse and Dialogue*, pages 74–83.

Yusuke Shinyama and Satoshi Sekine. 2003. Paraphrase acquisition for information extraction. In *Proceedings of Second International Workshop on Paraphrasing (IWP2003)*, pages 65–71.

Benjamin Snyder and Regina Barzilay. 2007. Database-text alignment via structured multilabel classification. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1713–1718.

J. Weeds and W. Weir. 2005. Co-occurrence retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):439–475.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammar. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*, Edinburgh, UK, August.

Open-Domain Semantic Role Labeling by Modeling Word Spans

Fei Huang
Temple University
1805 N. Broad St.
Wachman Hall 318
fei.huang@temple.edu

Alexander Yates
Temple University
1805 N. Broad St.
Wachman Hall 303A
yates@temple.edu

Abstract

Most supervised language processing systems show a significant drop-off in performance when they are tested on text that comes from a domain significantly different from the domain of the training data. Semantic role labeling techniques are typically trained on newswire text, and in tests their performance on fiction is as much as 19% worse than their performance on newswire text. We investigate techniques for building open-domain semantic role labeling systems that approach the ideal of a train-once, use-anywhere system. We leverage recently-developed techniques for learning representations of text using latent-variable language models, and extend these techniques to ones that provide the kinds of features that are useful for semantic role labeling. In experiments, our novel system reduces error by 16% relative to the previous state of the art on out-of-domain text.

1 Introduction

In recent semantic role labeling (SRL) competitions such as the shared tasks of CoNLL 2005 and CoNLL 2008, supervised SRL systems have been trained on newswire text, and then tested on both an in-domain test set (Wall Street Journal text) and an out-of-domain test set (fiction). All systems tested on these datasets to date have exhibited a significant drop-off in performance on the out-of-domain tests, often performing 15% worse or more on the fiction test sets. Yet the baseline from CoNLL 2005 suggests that the fiction texts are actually easier than the newswire texts. Such observations expose a weakness of current supervised natural language processing (NLP) technology for SRL: systems learn to identify semantic roles for the subset of language contained in the training data, but are not yet good at generalizing to language that has not been seen before.

We aim to build an *open-domain* supervised SRL system; that is, one whose performance on out-of-domain tests approaches the same level of performance as that of state-of-the-art systems on in-domain tests. Importantly, an open-domain system must not use any new labeled data beyond what is included in the original training text when running on a new domain. This allows the system to be ported to any new domain without any manual effort. In particular, it ought to apply to arbitrary Web documents, which are drawn from a huge variety of domains.

Recent theoretical and empirical evidence suggests that the fault for poor performance on out-of-domain tests lies with the representations, or sets of features, traditionally used in supervised NLP. Building on recent efforts in domain adaptation, we develop unsupervised techniques for learning new representations of text. Using latent-variable language models, we learn representations of texts that provide novel kinds of features to our supervised learning algorithms. Similar representations have proven useful in domain-adaptation for part-of-speech tagging and phrase chunking (Huang and Yates, 2009). We demonstrate how to learn representations that are effective for SRL. Experiments on out-of-domain test sets show that our learned representations can dramatically improve out-of-domain performance, and narrow the gap between in-domain and out-of-domain performance by half.

The next section provides background information on learning representations for NLP tasks using latent-variable language models. Section 3 presents our experimental setup for testing open-domain SRL. Sections 4, 5, 6 describe our SRL system: first, how we identify predicates in open-domain text, then how our baseline technique

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 968–978,
Uppsala, Sweden, 11-16 July 2010.

identifies and classifies arguments, and finally how we learn representations for improving argument identification and classification on out-of-domain text. Section 7 presents previous work, and Section 8 concludes and outlines directions for future work.

2 Open-Domain Representations Using Latent-Variable Language Models

Let $\mathcal{X}$ be an instance set for a learning problem; for SRL, this is the set of all (sentence,predicate) pairs. Let $\mathcal{Y}$ be the space of possible labels for an instance, and let f: $\mathcal{X} \rightarrow \mathcal{Y}$ be the target function to be learned. A *representation* is a function R: $\mathcal{X} \rightarrow \mathcal{Z}$, for some suitable feature space $\mathcal{Z}$ (such as $\mathbb{R}^{\mathrm{d}}$). A *domain* is defined as a distribution $\mathcal{D}$ over the instance set $\mathcal{X}$. An open-domain system observes a set of training examples $(R(x), f(x))$, where instances $x \in \mathcal{X}$ are drawn from a *source* domain, to learn a hypothesis for classifying examples drawn from a separate *target* domain.

Previous work by Ben-David *et al.* (2007; 2009) uses Vapnik-Chervonenkis (VC) theory to prove theoretical bounds on an open-domain learning machine's performance. Their analysis shows that the choice of representation is crucial to open-domain learning. As is customary in VC theory, a good choice of representation must allow a learning machine to achieve low error rates during training. Just as important, however, is that *the representation must simultaneously make the source and target domains look as similar to one another as possible*.

For open-domain SRL, then, the traditional representations are problematic. Typical representations in SRL and NLP use features of the local context to produce a representation. For instance, one dimension of a traditional representation R might be +1 if the instance contains the word "bank" as the head of a noun-phrase chunk that occurs before the predicate in the sentence, and 0 otherwise. Although many previous studies have shown that these features allow learning systems to achieve impressively low error rates during training, they also make texts from different domains look very dissimilar. For instance, a feature based on the word "bank" or "CEO" may be common in a domain of newswire text, but scarce or nonexistent in, say, biomedical literature.

In our recent work (Huang and Yates, 2009) we show how to build systems that learn new representations for open-domain NLP using latent-variable language models like Hidden Markov Models (HMMs). An HMM is a generative probabilistic model that generates each word x_i in the corpus conditioned on a latent variable Y_i. Each Y_i in the model takes on integral values from 1 to K, and each one is generated by the latent variable for the preceding word, Y_{i-1}. The distribution for a corpus $\mathbf{x} = (x_1, \ldots, x_N)$ and a set of state vectors $\mathbf{s} = (s_1, \ldots, s_N)$ is given by:

$$P(\mathbf{x}, \mathbf{s}) = \prod_i P(x_i|s_i)P(s_i|s_{i-1})$$

Using Expectation-Maximization (Dempster et al., 1977), it is possible to estimate the distributions for $P(x_i|s_i)$ and $P(s_i|s_{i-1})$ from unlabeled data. The Viterbi algorithm (Rabiner, 1989) can then be used to produce the optimal sequence of latent states s_i for a given instance $\mathbf{x}$. The output of this process is an integer (ranging from 1 to K) for every word x_i in the corpus. We use the integer value of s_i as a new feature for every x_i in the sentence.

In POS-tagging and chunking experiments, these learned representations have proven to meet both of Ben-David *et al.*'s criteria for open-domain representations: first, they are useful in making predictions on the training text because the HMM latent states categorize tokens according to distributional similarity. And second, it would be difficult to tell two domains apart based on their HMM labels, since the same HMM state can generate similar words from a variety of domains. In what follows, we adapt these representation-learning concepts to open-domain SRL.

3 Experimental Setup

We test our open-domain semantic role labeling system using data from the CoNLL 2005 shared task (Carreras and Màrquez, 2005). We use the standard training set, consisting of sections 02-21 of the Wall Street Journal (WSJ) portion of the Penn Treebank, labeled with PropBank (Palmer et al., 2005) annotations for predicates and arguments. We perform our tests on the Brown corpus (Kucera and Francis, 1967) test data from CoNLL 2005, consisting of 3 sections (ck01-ck03) of propbanked Brown corpus data. This test set consists of 426 sentences containing 7,159 tokens, 804 propositions, and 2,177 arguments. While the

training data contains newswire text, the test sentences are drawn from the domain of "general fiction," and contain an entirely different style (or styles) of English. The data also includes a second test set of in-domain text (section 23 of the Treebank), which we refer to as the WSJ test set and use as a reference point.

Every sentence in the dataset is automatically annotated with a number of NLP pipeline systems, including part-of-speech (POS) tags, phrase chunk labels (Carreras and Màrquez, 2003), named-entity tags, and full parse information by multiple parsers. These pipeline systems are important for generating features for SRL, and one key reason for the poor performance of SRL systems on the Brown corpus is that the pipeline systems themselves perform worse. The Charniak parser, for instance, drops from an F1 of 88.25 on the WSJ test to a F1 of 80.84 on the Brown corpus. For the chunker and POS tagger, the drop-offs are less severe: 94.89 to 91.73, and 97.36 to 94.73.

Toutanova *et al.* (2008) currently have the best-performing SRL system on the Brown corpus test set with an F1 score of 68.81 (80.8 for the WSJ test). They use a discriminative reranking approach to jointly predict the best set of argument boundaries and the best set of argument labels for a predicate. Like the best systems from the CoNLL 2005 shared task (Punyakanok et al., 2008; Pradhan et al., 2005), they also use features from multiple parses to remain robust in the face of parser error. Owing to the established difficulty of the Brown test set and the different domains of the Brown test and WSJ training data, this dataset makes for an excellent testbed for open-domain semantic role labeling.

4 Predicate Identification

In order to perform true open-domain SRL, we must first consider a task which is not formally part of the CoNLL shared task: the task of identifying predicates in a given sentence. While this task is almost trivial in the WSJ test set, where all but two out of over 5000 predicates can be observed in the training data, it is significantly more difficult in an open-domain setting. In the Brown test set, 6.1% of the predicates do not appear in the training data, and 11.8% of the predicates appear at most twice in the training data (*c.f.* 1.5% of the WSJ test predicates that appear at most twice in training). In addition, many words which appear as predicates in training may not be predicates in the test set. In an open-domain setting, therefore, we cannot rely solely on a catalog of predicates from the training data.

	Baseline			HMM		
Freq	P	R	F1	P	R	F1
0	89.1	80.4	84.5	93.5	84.3	**88.7**
0-2	87.4	84.7	86.0	91.6	88.8	**90.2**
all	87.8	92.5	90.1	90.8	96.3	**93.5**

Table 1: Using HMM features in predicate identification reduces error in out-of-domain tests by 34.3% overall, and by 27.1% for OOV predicates. "Freq" refers to frequency in the training data. There were 831 predicates in total; 51 never appeared in training and 98 appeared at most twice.

To address the task of open-domain predicate identification, we construct a Conditional Random Field (CRF) (Lafferty et al., 2001) model with target labels of B-Pred, I-Pred, and O-Pred (for the beginning, interior, and outside of a predicate). We use an open source CRF software package to implement our CRF models.[1] We use words, POS tags, chunk labels, and the predicate label at the preceding and following nodes as features for our Baseline system. To learn an open-domain representation, we then trained an 80 state HMM on the unlabeled texts of the training and Brown test data, and used the Viterbi optimum states of each word as categorical features.

The results of our Baseline and HMM systems appear in Table 1. For predicates that never or rarely appear in training, the HMM features increase F1 by 4.2, and they increase the overall F1 of the system by 3.5 to 93.5, which approaches the F1 of 94.7 that the Baseline system achieves on the in-domain WSJ test set. Based on these results, we were satisfied that our system could find predicates in open-domain text. In all subsequent experiments, we fall back on the standard evaluation in which it is assumed that the boundaries of the predicate are given. This allows us to compare with previous work.

5 Semantic Role Labeling with HMM-based Representations

Following standard practice, we divide the SRL task into two parts: argument identification and

[1] Available from http://sourceforge.net/projects/crf/

argument classification. We treat both sub-tasks as sequence-labeling problems. During argument identification, the system must label each token with labels that indicate either the beginning or interior of an argument (B-Arg or I-Arg), or a label that indicates the token is not part of an argument (O-Arg). During argument classification, the system labels each token that is part of an argument with a class label, such as Arg0 or ArgM. Following argument classification, multi-word arguments may have different classification labels for each token. We post-process the labels by changing them to match the label of the first token. We use CRFs as our models for both tasks (Cohn and Blunsom, 2005).

Most previous approaches to SRL have relied heavily on parsers, and especially constituency parsers. Indeed, when SRL systems use gold standard parses, they tend to perform extremely well (Toutanova et al., 2008). However, as several previous studies have noted (Gildea, 2001; Pradhan et al., 2007), using parsers can cause problems for open-domain SRL. The parsers themselves may not port well to new domains, or the features they generate for SRL may not be stable across domains, and therefore may cause sparse data problems on new domains. Our first step is therefore to build an SRL system that relies on partial parsing, as was done in CoNLL 2004 (Carreras and Màrquez, 2004). We then gradually add in less-sparse alternatives for the syntactic features that previous systems derive from parse trees.

During argument identification we use the features below to predict the label A_i for token w_i:

- *words*: w_i, w_{i-1}, and w_{i+1}
- *parts of speech (POS)*: POS tags t_i, t_{i-1}, and t_{i+1}
- *chunk labels*: (*e.g.*, B-NP, I-VP, or O) chunk tags c_i, c_{i-1}, and c_{i+1}
- *combinations*: $c_i t_i$, $t_i w_i$, $c_i t_i w_i$
- *NE*: the named entity type n_i of w_i
- *position*: whether the word occurs before or after the predicate
- *distance*: the number of intervening tokens between w_i and the target predicate
- *POS before, after predicate*: the POS tag of the tokens immediately preceding and following the predicate
- *Chunk before, after predicate*: the chunk type of the tokens immediately preceding and following the predicate
- *Transition*: for prediction node A_i, we use $A_{i-1}$and A_{i+1} as features

For argument classification, we add the features below to those listed above:

- *arg ID*: the labels A_i produced by arg. identification (B-Arg, I-Arg, or O)
- *combination*: predicate + first argument word, predicate+ last argument word, predicate + first argument POS, predicate + last argument POS
- *head distance*: the number of tokens between the first token of the argument phrase and the target predicate
- *neighbors*: the words immediately before and after the argument.

We refer to the CRF model with these features as our Baseline SRL system; in what follows we extend the Baseline model with more sophisticated features.

5.1 Incorporating HMM-based Representations

As a first step towards an open-domain representation, we use an HMM with 80 latent state values, trained on the unlabeled text of the training and test sets, to produce Viterbi-optimal state values s_i for every token in the corpus. We then add the following features to our CRFs for both argument identification and classification:

- *HMM states*: HMM state values s_i, s_{i-1}, and s_{i+1}
- *HMM states before, after predicate*: the state value of the tokens immediately preceding and following the predicate

We call the resulting model our Baseline+HMM system.

5.2 Path Features

Despite all of the features above, the SRL system has very little information to help it determine the syntactic relationship between a target predicate and a potential argument. For instance, these baseline features provide only crude distance information to distinguish between multiple arguments that follow a predicate, and they make it difficult to correctly identify clause arguments or arguments that appear far from the predicate. Our system needs features that can help distinguish between different syntactic relationships, without being overly sensitive to the domain.

As a step in this direction, we introduce *path* features: features for the sequence of tokens be-

System	P	R	F1
Baseline	63.9	59.7	61.7
Baseline+HMM	68.5	62.7	65.5
Baseline+HMM+Paths	70.0	65.6	67.7
Toutanova *et al.* (2008)	NR	NR	**68.8**

Table 2: Naïve path features improve our baseline, but not enough to match the state-of-the-art. Toutanova *et al.* do not report (NR) separate values for precision and recall on this dataset. Differences in both precision and recall between the baseline and the other systems are statistically significant at $p < 0.01$ using the two-tailed Fisher's exact test.

tween a predicate and a potential argument. In standard SRL systems, these path features usually consist of a sequence of constituent parse nodes representing the shortest path through the parse tree between a word and the predicate (Gildea and Jurafsky, 2002). We substitute paths that do not depend on parse trees. We use four types of paths: word paths, POS paths, chunk paths, and HMM state paths. Given an input sentence labeled with POS tags, and chunks, we construct path features for a token w_i by concatenating words (or tags or chunk labels) between w_i and the predicate. For example, in the sentence "The HIV infection rate is expected to peak in 2010," the word path between "rate" and predicate "peak" would be "is expected to", and the POS path would be "VBZ VBD TO."

Since word, POS, and chunk paths are all subject to data sparsity for arguments that are far from the predicate, we build less-sparse path features by using paths of HMM states. If we use a reasonable number of HMM states, each category label is much more common in the training data than the average word, and paths containing the HMM states should be much less sparse than word paths, and even chunk paths. In our experiments, we use 80-state HMMs.

We call the result of adding path features to our feature set the Baseline+HMM+Paths system((BL). Table 2 shows the performance of our three baseline systems. In this open-domain SRL experiment, path features improve over the Baseline's F1 by 6 points, and by 2.2 points over Baseline+HMM, although the improvement is not enough to match the state-of-the-art system by Toutanova *et al.*

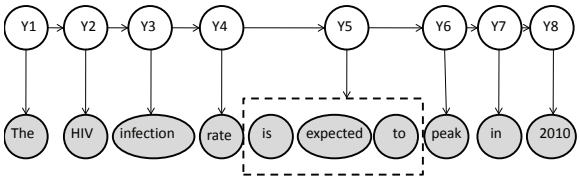


Figure 1: The Span-HMM over the sentence. It shows the span of length 3.

6 Representations for Word Spans

Despite partial success in improving our baseline SRL system with path features, these features still suffer from data sparsity — many paths in the test set are never or very rarely observed during training, so the CRF model has little or no data points from which to estimate accurate parameters for these features. In response, we introduce latent variable models of *word spans*, or sequences of words. As with the HMM models above, the latent states for word spans can be thought of as probabilistic categories for the spans. And like the HMM models, we can turn the word span models into representations by using the state value for a span as a feature in our supervised SRL system. Unlike path features, the features from our models of word spans consist of a single latent state value rather than a concatenation of state values, and as a consequence they tend to be much less sparse in the training data.

6.1 Span-HMM Representations

We build our latent-variable models of word spans using variations of Hidden Markov Models, which we call Span-HMMs. Figure 1 shows a graphical model of a Span-HMM. Each Span-HMM behaves just like a regular HMM, except that it includes one node, called a *span node*, that can generate an entire span rather than a single word. For instance, in the Span-HMM of Figure 1, node y_5 is a span node that generates a span of length 3: "is expected to."

Span-HMMs can be used to provide a single categorical value for any span of a sentence using the usual Viterbi algorithm for HMMs. That is, at test time, we generate a Span-HMM feature for word w_j by constructing a Span-HMM that has a span node for the sequence of words between w_j and the predicate. We determine the Viterbi optimal state of this span node, and use that state as the value of the new feature. In our example in Figure 1, the value of span node y_5 is used as a feature for

the token "rate", since y_5 generates the sequence of words between "rate" and the predicate "peak."

Notice that by using Span-HMMs to provide these features, we have condensed all paths in our data into a small number of categorical values. Whereas there are a huge number of variations to the spans themselves, we can constrain the number of categories for the Span-HMM states to a reasonable number such that each category is likely to appear often in the training data. The value of each Span-HMM state then represents a cluster of spans with similar delimiting words; some clusters will correlate with spans between predicates and arguments, and others with spans that do not connect predicates and arguments. As a result, Span-HMM features are not sparse, and they correlate with the target function, making them useful in learning an SRL model.

6.2 Parameter Estimation

We use a variant of the Baum-Welch algorithm to train our Span-HMMs on unlabeled text. In order for this to work, we need to provide Baum-Welch with a modified view of the data so that span nodes can generate multiple consecutive words in a sentence. First, we take every sentence S in our training data and generate the set $Spans(S)$ of all valid spans in the sentence. For efficiency's sake, we use only spans of length less than 15; approximately 95% of the arguments in our dataset were within 15 words of the predicate, so even with this restriction we are able to supply features for nearly all valid arguments. The second step of our training procedure is to create a separate data point for each span of S. For each span $t \in Spans(S)$, we construct a Span-HMM with a regular node generating each element of S, except that a span node generates all of t. Thus, our training data contains many different copies of each sentence S, with a different Span-HMM generating each copy.

Intuitively, running Baum-Welch over this data means that a span node with state k will be likely to generate two spans t_1 and t_2 if t_1 and t_2 tend to appear in similar contexts. That is, they should appear between words that are also likely to be generated by the same latent state. Thus, certain values of k will tend to appear for spans between predicates and arguments, and others will tend to appear between predicates and non-arguments. This makes the value k informative for both argument identification and argument classification.

6.3 Memory Considerations

Memory usage is a major issue for our Span-HMM models. We represent emission distributions as multinomials over discrete observations. Since there are millions of different spans in our data, a straightforward implementation would require millions of parameters for each latent state of the Span-HMM.

We use two related techniques to get around this problem. In both cases, we use a second HMM model, which we call the base HMM to distinguish from our Span-HMM, to back-off from the explicit word sequence. We use the largest number of states for HMMs that can be fit into memory. Let S be a sentence, and let $\hat{s}$ be the sequence of optimal latent state values for S produced by our base HMM. Our first approach trains the Span-HMM on $Spans(\hat{s})$, rather than $Spans(S)$. If we use a small enough number of latent states in the base HMM (in experiments, we use 10 latent states), we drastically reduce the number of different spans in the data set, and therefore the number of parameters required for our model. We call this representation Span-HMM-Base10. As with our other HMM-based models, we use the largest number of latent states that will allow the resulting model to fit in our machine's memory — our previous experiments on representations for part-of-speech tagging suggest that more latent states are usually better.

While our first technique solves the memory issue, it also loses some of the power of our original Span-HMM model by using a very coarse-grained base HMM clustering of the text into 10 categories. Our second approach trains a separate Span-HMM model for spans of different lengths. Since we need only one model in memory at a time, this allows each one to consume more memory. We therefore use base HMM models with more latent states (up to 20) to annotate our sentences, and then train on the resulting $Spans(\hat{s})$ as before. With this technique, we produce features that are combinations of the state value for span nodes and the length of the span, in order to indicate which of our Span-HMM models the state value came from. We call this representation Span-HMM-BaseByLength.

6.4 Combining Multiple Span-HMMs

So far, our Span-HMM models produce one new feature for every token during argument identifi-

System	P	R	F1
Baseline+HMM+Paths	70.0	65.6	67.7
Toutanova *et al.*	NR	NR	68.8
Span-HMM-Base10	74.5	69.3	71.8
Span-HMM-BaseByLength	76.3	70.2	73.1
Multi-Span-HMM	**77.0**	**70.9**	**73.8**

Table 3: Span-HMM features significantly improve over state-of-the-art results in out-of-domain SRL. Differences in both precision and recall between the baseline and the Span-HMM systems are statistically significant at $p < 0.01$ using the two-tailed Fisher's exact test.

System	WSJ	Brown	Diff
Multi-Span-HMM	79.2	**73.8**	**5.4**
Toutanova *et al.* (2008)	**80.8**	68.8	12.0
Pradhan *et al.* (2005)	78.6	68.4	10.2
Punyakanok *et al.* (2008)	79.4	67.8	11.6

Table 4: Multi-Span-HMM has a much smaller drop-off in F1 than comparable systems on out-of-domain test data vs in-domain test data.

cation and classification. While these new features may be very helpful, ideally we would like our learned representations to produce multiple useful features for the CRF model, so that the CRF can combine the signals from each feature to learn a sophisticated model. Towards this goal, we train N independent versions of our Span-HMM-BaseByLength models, each with a random initialization for the Baum-Welch algorithm. Since Baum-Welch is a hill-climbing algorithm, it should find local, but not necessarily global, optima for the parameters of each Span-HMM-BaseByLength model. When we decode each of the models on training and test texts, we will obtain N different sequences of latent states, one for each locally-optimized model. Thus we obtain N different, independent sources of features. We call the CRF model with these N Span-HMM features the Multi-Span-HMM model(MSH); in experiments we use $N = 5$.

6.5 Results and Discussion

Results for the Span-HMM models on the CoNLL 2005 Brown corpus are shown in Table 3. All three versions of the Span-HMM outperform Toutanova *et al.*'s system on the Brown corpus, with the Multi-Span-HMM gaining 5 points in F1. The Multi-Span-HMM model improves over the Baseline+HMM+Paths model by 7 points in precision, and 5.3 points in recall. Among the Span-HMM models, the use of more states in the Span-HMM-BaseByLength model evidently outweighed the cost of splitting the model into separate versions for different length spans. Using multiple independent copies of the Span-HMMs provides a small (0.7) gain in precision and recall. Differences among the different Span-HMM models were not statistically significant, except that the difference in precision between the Multi-Span-HMM and the Span-HMM-Base10 is significant at $p < .1$.

Table 4 shows the performance drop-off for top SRL systems when applied to WSJ test data and Brown corpus test data. The Multi-Span-HMM model performs near the state-of-the-art on the WSJ test set, and its F1 on out-of-domain data drops only about half as much as comparable systems. Note that several of the techniques used by other systems, such as using features from k-best parses or jointly modeling the dependencies among arguments, are complementary to our techniques, and may boost the performance of our system further.

Table 5 breaks our results down by argument type. Most of our improvement over the Baseline system comes from the core arguments A0 and A1, but also from a few adjunct types like AM-TMP and AM-LOC. Figure 2 shows that when the argument is close to the predicate, both systems perform well, but as the distance from the predicate grows, our Multi-Span-HMM system is better able to identify and classify arguments than the Baseline+HMM+Paths system.

Table 6 provides results for argument identification and classification separately. As Pradhan *et al.*previously showed (Pradhan et al., 2007), SRL systems tend to have an easier time with porting argument identification to new domains, but are less strong at argument classification on new domains. Our baseline system decreases in F-score from 81.5 to 78.9 for argument identification, but suffers a much larger 8% drop in argument classification. The Multi-Span-HMM model improves over the Baseline in both tasks and on both test sets, but the largest improvement (6%) is in argument classification on the Brown test set.

To help explain the success of the Span-HMM techniques, we measured the sparsity of our path

	Overall	A0	A1	A2	A3	A4	ADV	DIR	DIS	LOC	MNR	MOD	NEG	PNC	TMP	R-A0	R-A1
Num	2177	566	676	147	12	15	143	53	22	85	110	91	50	17	112	25	21
BL	67.7	76.2	70.6	64.8	59.0	71.2	52.7	54.8	71.9	67.5	58.3	90.9	90.0	50.0	76.5	76.5	71.3
MSH	73.8	82.5	73.6	63.9	60.3	73.3	50.8	52.9	70.0	70.3	52.7	94.2	92.9	51.6	81.6	84.4	75.7

Table 5: SRL results (F1) on the Brown test corpus broken down by role type. BL is the Baseline+HMM+Paths model, MSH is the Multi-Span-HMM model. Column 8 to 16 are all adjuncts (AM-). We omit roles with ten or fewer examples.

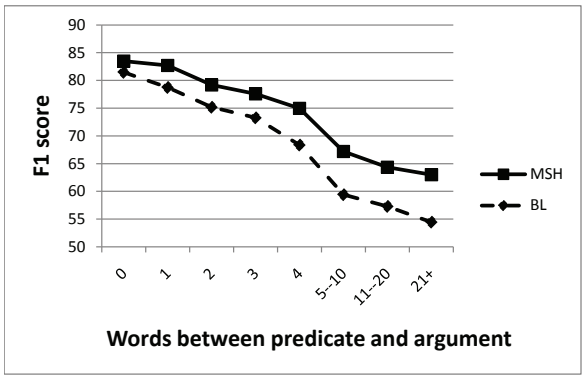


Figure 2: The Multi-Span-HMM (MSH) model is better able to identify and classify arguments that are far from the predicate than the Baseline+HMM+Paths (BL) model.

	Test	Id.F1	Accuracy
BL	WSJ	81.5	93.7
	Brown	78.9	85.8
MSH	WSJ	83.9	94.4
	Brown	80.3	91.9

Table 6: Baseline (BL) and Multi-Span-HMM (MSH) performance on argument identification (Id.F1) and argument classification.

and Span-HMM features. Figure 3 shows the percentage of feature values in the Brown corpus that appear more than twice, exactly twice, or exactly once in the training data. While word path features can be highly valuable when there is training data available for them, only about 11% of the word paths in the Brown test set also appeared at all in the training data. POS and chunk paths fared a bit better (22% and 33% respectively), but even then nearly 70% of all feature values had no available training data. HMM and Span-HMM-Base10 paths achieved far better success in this respect. Importantly, the improvement is mostly due to features that are seen *often* in training, rather than features that were seen just once or twice. Thus Span-

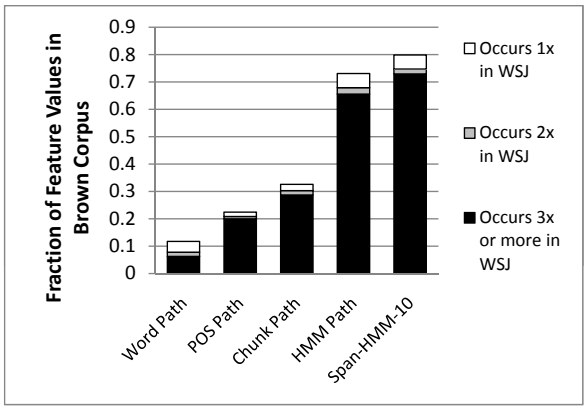


Figure 3: HMM path and Span-HMM features are far more likely to appear often in training data than the word, POS, and chunk path features. Over 70% of Span-HMM-Base10 features in the Brown corpus appear at least three times during training; in contrast, fewer than 33% of chunk path features in the Brown corpus appear at all during training.

HMMs derive their power as representations for open-domain SRL from the fact that they provide features that are mostly the same across domains; 80% of the features of our Span-HMM-Base10 in the Brown corpus were observed at least once in the training data.

Table 7 shows examples of spans that were clustered into the same Span-HMM state, along with word to either side. All four examples are cases where the Span-HMM-Base10 model correctly tagged the following argument, but the Baseline+HMM+Paths model did not. We can see that the paths of these four examples are completely different, but the words surrounding them are very similar. The emission from a span node are very sparse, so the Span-HMM has unsurprisingly learned to cluster spans according to the HMM states that precede and follow the span node. This is by design, as this kind of distributional clustering is helpful for identifying and classifying arguments. One potentially interesting

Predicate	Span	B-Arg
picked	the things up	from
passed	through the barbed wire	at
come	down from Sundays	to
sat	over his second rock	in

Table 7: Example spans labeled with the same Span-HMM state. The examples are taken from sentences where the Span-HMM-Base10 model correctly identified the argument on the right, but the Baseline+HMM+Paths model did not.

question for future work is whether a less sparse model of the spans themselves, such as a Naïve Bayes model for the span node, would yield a better clustering for producing features for semantic role labeling.

7 Previous Work

Deschact and Moens (2009) use a latent-variable language model to provide features for an SRL system, and they show on CoNLL 2008 data that they can significantly improve performance when little labeled training data is available. They do not report on out-of-domain tests. They use HMM language models trained on unlabeled text, much like we use in our baseline systems, but they do not consider models of word spans, which we found to be most beneficial. Downey *et al.* (2007b) also incorporate HMM-based representations into a system for the related task of Web information extraction, and are able to show that the system improves performance on rare terms.

Fürstenau and Lapata (2009b; 2009a) use semi-supervised techniques to automatically annotate data for previously unseen predicates with semantic role information. This task differs from ours in that it focuses on previously unseen predicates, which may or may not be part of text from a new domain. Their techniques also result in relatively lower performance (F1 between 15 and 25), although their tests are on a more difficult and very different corpus. Weston *et al.* (2008) use deep learning techniques based on semi-supervised embeddings to improve an SRL system, though their tests are on in-domain data. Unsupervised SRL systems (Swier and Stevenson, 2004; Grenager and Manning, 2006; Abend et al., 2009) can naturally be ported to new domains with little trouble, but their accuracy thus far falls short of state-of-the-art supervised and semi-supervised systems.

The disparity in performance between in-domain and out-of-domain tests is by no means restricted to SRL. Past research in a variety of NLP tasks has shown that parsers (Gildea, 2001), chunkers (Huang and Yates, 2009), part-of-speech taggers (Blitzer et al., 2006), named-entity taggers (Downey et al., 2007a), and word sense disambiguation systems (Escudero et al., 2000) all suffer from a similar drop-off in performance on out-of-domain tests. Numerous domain adaptation techniques have been developed to address this problem, including self-training (McClosky et al., 2006) and instance weighting (Bacchiani et al., 2006) for parser adaptation and structural correspondence learning for POS tagging (Blitzer et al., 2006). Of these techniques, structural correspondence learning is closest to our technique in that it is a form of representation learning, but it does not learn features for word spans. None of these techniques have been successfully applied to SRL.

8 Conclusion and Future Work

We have presented novel representation-learning techniques for building an open-domain SRL system. By incorporating learned features from HMMs and Span-HMMs trained on unlabeled text, our SRL system is able to correctly identify predicates in out-of-domain text with an F1 of 93.5, and it can identify and classify arguments to predicates with an F1 of 73.8, outperforming comparable state-of-the-art systems. Our successes so far on out-of-domain tests bring hope that supervised NLP systems may eventually achieve the ideal where they no longer need new manually-labeled training data for every new domain. There are several potential avenues for further progress towards this goal, including the development of more portable SRL pipeline systems, and especially parsers. Developing techniques that can incrementally adapt to new domains without the computational expense of retraining the CRF model every time would help make open-domain SRL more practical.

Acknowledgments

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

References

Omri Abend, Roi Reichart, and Ari Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the ACL*.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 20*, Cambridge, MA. MIT Press.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. 2009. A theory of learning from different domains. *Machine Learning*, (to appear).

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

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

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Trevor Cohn and Phil Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL*.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

D. Downey, M. Broadhead, and O. Etzioni. 2007a. Locating complex named entities in web text. In *Procs. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007b. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*.

G. Escudero, L. Márquez, and G. Rigau. 2000. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *EMNLP/VLC*.

Hagen Fürstenau and Mirella Lapata. 2009a. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11–20.

Hagen Fürstenau and Mirella Lapata. 2009b. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 220–228.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea. 2001. Corpus Variation and Parser Performance. In *Conference on Empirical Methods in Natural Language Processing*.

Trond Grenager and Christopher D Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

H. Kucera and W.N. Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Press.

J. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 337–344.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics Journal*, 31(1).

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

Sameer Pradhan, Wayne Ward, and James H. Martin. 2007. Towards robust semantic role labeling. In *Proceedings of NAACL-HLT*, pages 556–563.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Jason Weston, Frederic Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*.

Learning Script Knowledge with Web Experiments

Michaela Regneri **Alexander Koller** **Manfred Pinkal**
Department of Computational Linguistics and Cluster of Excellence
Saarland University, Saarbrücken
{regneri|koller|pinkal}@coli.uni-saarland.de

Abstract

We describe a novel approach to unsupervised learning of the events that make up a script, along with constraints on their temporal ordering. We collect natural-language descriptions of script-specific event sequences from volunteers over the Internet. Then we compute a graph representation of the script's temporal structure using a multiple sequence alignment algorithm. The evaluation of our system shows that we outperform two informed baselines.

1 Introduction

A script is "a standardized sequence of events that describes some stereotypical human activity such as going to a restaurant or visiting a doctor" (Barr and Feigenbaum, 1981). Scripts are fundamental pieces of commonsense knowledge that are shared between the different members of the same culture, and thus a speaker assumes them to be tacitly understood by a hearer when a scenario related to a script is evoked: When one person says "I'm going shopping", it is an acceptable reply to say "did you bring enough money?", because the SHOPPING script involves a 'payment' event, which again involves the transfer of money.

It has long been recognized that text understanding systems would benefit from the implicit information represented by a script (Cullingford, 1977; Mueller, 2004; Miikkulainen, 1995). There are many other potential applications, including automated storytelling (Swanson and Gordon, 2008), anaphora resolution (McTear, 1987), and information extraction (Rau et al., 1989).

However, it is also commonly accepted that the large-scale manual formalization of scripts is infeasible. While there have been a few attempts at doing this (Mueller, 1998; Gordon, 2001), efforts in which expert annotators create script knowledge bases clearly don't scale. The same holds true of the script-like structures called "scenario frames" in FrameNet (Baker et al., 1998).

There has recently been a surge of interest in automatically learning script-like knowledge resources from corpora (Chambers and Jurafsky, 2008b; Manshadi et al., 2008); but while these efforts have achieved impressive results, they are limited by the very fact that a lot of scripts – such as SHOPPING – are shared implicit knowledge, and their events are therefore rarely elaborated in text.

In this paper, we propose a different approach to the unsupervised learning of script-like knowledge. We focus on the temporal event structure of scripts; that is, we aim to learn what phrases can describe the same event in a script, and what constraints must hold on the temporal order in which these events occur. We approach this problem by asking non-experts to describe typical event sequences in a given scenario over the Internet. This allows us to assemble large and varied collections of event sequence descriptions (ESDs), which are focused on a single scenario. We then compute a *temporal script graph* for the scenario by identifying corresponding event descriptions using a Multiple Sequence Alignment algorithm from bioinformatics, and converting the alignment into a graph. This graph makes statements about what phrases can describe the same event of a scenario, and in what order these events can take place. Crucially, our algorithm exploits the sequential structure of the ESDs to distinguish event descriptions that occur at different points in the script storyline, even when they are semantically similar. We evaluate our script graph algorithm on ten unseen scenarios, and show that it significantly outperforms a clustering-based baseline.

The paper is structured as follows. We will first position our research in the landscape of related work in Section 2. We will then define how

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 979–988,
Uppsala, Sweden, 11-16 July 2010.

we understand scripts, and what aspect of scripts we model here, in Section 3. Section 4 describes our data collection method, and Section 5 explains how we use Multiple Sequence Alignment to compute a temporal script graph. We evaluate our system in Section 6 and conclude in Section 7.

2 Related Work

Approaches to learning script-like knowledge are not new. For instance, Mooney (1990) describes an early attempt to acquire causal chains, and Smith and Arnold (2009) use a graph-based algorithm to learn temporal script structures. However, to our knowledge, such approaches have never been shown to generalize sufficiently for wide coverage application, and none of them was rigorously evaluated.

More recently, there have been a number of approaches to automatically learning event chains from corpora (Chambers and Jurafsky, 2008b; Chambers and Jurafsky, 2009; Manshadi et al., 2008). These systems typically employ a method for classifying temporal relations between given event descriptions (Chambers et al., 2007; Chambers and Jurafsky, 2008a; Mani et al., 2006). They achieve impressive performance at extracting high-level descriptions of procedures such as a CRIMINAL PROCESS. Because our approach involves directly asking people for event sequence descriptions, it can focus on acquiring specific scripts from arbitrary domains, and we can control the level of granularity at which scripts are described. Furthermore, we believe that much information about scripts is usually left implicit in texts and is therefore easier to learn from our more explicit data. Finally, our system automatically learns different phrases which describe the same event together with the temporal ordering constraints.

Jones and Thompson (2003) describe an approach to identifying different natural language realizations for the same event considering the temporal structure of a scenario. However, they don't aim to acquire or represent the temporal structure of the whole script in the end.

In its ability to learn paraphrases using Multiple Sequence Alignment, our system is related to Barzilay and Lee (2003). Unlike Barzilay and Lee, we do not tackle the general paraphrase problem, but only consider whether two phrases describe the same event in the context of the same script. Furthermore, the atomic units of our alignment process are entire phrases, while in Barzilay and Lee's setting, the atomic units are words.

Finally, it is worth pointing out that our work is placed in the growing landscape of research that attempts to learn linguistic information out of data directly collected from users over the Internet. Some examples are the general acquisition of commonsense knowledge (Singh et al., 2002), the use of browser games for that purpose (von Ahn and Dabbish, 2008), and the collaborative annotation of anaphoric reference (Chamberlain et al., 2009). In particular, the use of the Amazon Mechanical Turk, which we use here, has been evaluated and shown to be useful for language processing tasks (Snow et al., 2008).

3 Scripts

Before we delve into the technical details, let us establish some terminology. In this paper, we distinguish *scenarios*, as classes of human activities, from *scripts*, which are stereotypical models of the internal structure of these activities. Where EATING IN A RESTAURANT is a scenario, the script describes a number of events, such as ordering and leaving, that must occur in a certain order in order to constitute an EATING IN A RESTAURANT activity. The classical perspective on scripts (Schank and Abelson, 1977) has been that next to defining some events with temporal constraints, a script also defines their participants and their causal connections.

Here we focus on the narrower task of learning the events that a script consists of, and of modeling and learning the temporal ordering constraints that hold between them. Formally, we will specify a script (in this simplified sense) in terms of a directed graph $G_s = (E_s, T_s)$, where E_s is a set of nodes representing the events of a scenario s, and T_s is a set of edges (e_i, e_k) indicating that the event e_i typically happens before e_k in s. We call G_s the *temporal script graph (TSG)* for s.

Each event in a TSG can usually be expressed with many different natural-language phrases. As the TSG in Fig. 3 illustrates, the first event in the script for EATING IN A FAST FOOD RESTAURANT can be equivalently described as 'walk to the counter' or 'walk up to the counter'; even phrases like 'walk into restaurant', which would not usually be taken as paraphrases of these, can be accepted as describing the same event in the context

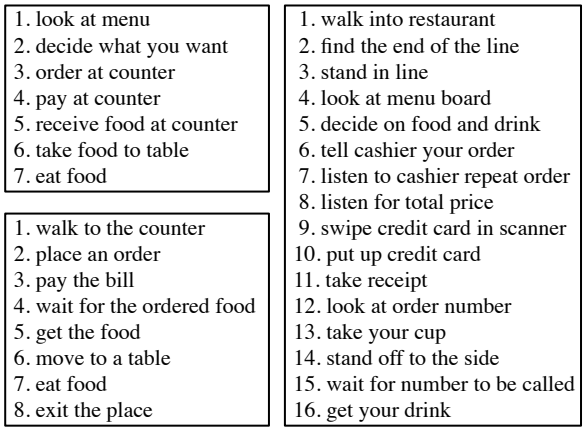


Figure 1: Three event sequence descriptions

of this scenario. We call a natural-language realization of an individual event in the script an *event description*, and we call a sequence of event descriptions that form one particular instance of the script an *event sequence description (ESD)*. Examples of ESDs for the FAST FOOD RESTAURANT script are shown in Fig. 1.

One way to look at a TSG is thus that its nodes are equivalence classes of different phrases that describe the same event; another is that valid ESDs can be generated from a TSG by randomly selecting phrases from some nodes and arranging them in an order that respects the temporal precedence constraints in T_s. Our goal in this paper is to take a set of ESDs for a given scenario as our input and then compute a TSG that clusters different descriptions of the same event into the same node, and contains edges that generalize the temporal information encoded in the ESDs.

4 Data Acquisition

In order to automatically learn TSGs, we selected 22 scenarios for which we collect ESDs. We deliberately included scenarios of varying complexity, including some that we considered hard to describe (CHILDHOOD, CREATE A HOMEPAGE), scenarios with highly variable orderings between events (MAKING SCRAMBLED EGGS), and scenarios for which we expected cultural differences (WEDDING).

We used the Amazon Mechanical Turk[1] to collect the data. For every scenario, we asked 25 people to enter a typical sequence of events in this scenario, in temporal order and in "bullet point style". We required the annotators to enter at least 5 and at most 16 events. Participants were allowed to skip a scenario if they felt unable to enter events for it, but had to indicate why. We did not restrict the participants (e.g. to native speakers).

In this way, we collected 493 ESDs for the 22 scenarios. People used the possibility to skip a form 57 times. The most frequent explanation for this was that they didn't know how a certain scenario works: The scenario with the highest proportion of skipped forms was CREATE A HOMEPAGE, whereas MAKING SCRAMBLED EGGS was the only one in which nobody skipped a form. Because we did not restrict the participants' inputs, the data was fairly noisy. For the purpose of this study, we manually corrected the data for orthography and filtered out forms that were written in broken English or did not comply with the task (e.g. when users misunderstood the scenario, or did not list the event descriptions in temporal order). Overall we discarded 15% of the ESDs.

Fig. 1 shows three of the ESDs we collected for EATING IN A FAST-FOOD RESTAURANT. As the example illustrates, descriptions differ in their starting points ('walk into restaurant' vs. 'walk to counter'), the granularity of the descriptions ('pay the bill' vs. event descriptions 8–11 in the third sequence), and the events that are mentioned in the sequence (not even 'eat food' is mentioned in all ESDs). Overall, the ESDs we collected consisted of 9 events on average, but their lengths varied widely: For most scenarios, there were significant numbers of ESDs both with the minimum length of 5 and the maximum length of 16 and everything in between. Combined with the fact that 93% of all individual event descriptions occurred only once, this makes it challenging to align the different ESDs with each other.

5 Temporal Script Graphs

We will now describe how we compute a temporal script graph out of the collected data. We proceed in two steps. First, we identify phrases from different ESDs that describe the same event by computing a Multiple Sequence Alignment (MSA) of all ESDs for the same scenario. Then we postprocess the MSA and convert it into a temporal script graph, which encodes and generalizes the temporal information contained in the original ESDs.

[1] http://www.mturk.com/

row	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$
1	⊘	walk into restaurant	⊘	enter restaurant
2	⊘	⊘	walk to the counter	go to counter
3	⊘	find the end of the line	⊘	⊘
4	⊘	stand in line	⊘	⊘
5	look at menu	look at menu board	⊘	⊘
6	decide what you want	decide on food and drink	⊘	make selection
7	order at counter	tell cashier your order	place an order	place order
8	⊘	listen to cashier repeat order	⊘	⊘
9	pay at counter	⊘	pay the bill	pay for food
10	⊘	listen for total price	⊘	⊘
11	⊘	swipe credit card in scanner	⊘	⊘
12	⊘	put up credit card	⊘	⊘
13	⊘	take receipt	⊘	⊘
14	⊘	look at order number	⊘	⊘
15	⊘	take your cup	⊘	⊘
16	⊘	stand off to the side	⊘	⊘
17	⊘	wait for number to be called	wait for the ordered food	⊘
18	receive food at counter	get your drink	get the food	pick up order
19	⊘	⊘	⊘	pick up condiments
20	take food to table	⊘	move to a table	go to table
21	eat food	⊘	eat food	consume food
22	⊘	⊘	⊘	clear tray
22	⊘	⊘	exit the place	⊘

Figure 2: A MSA of four event sequence descriptions

5.1 Multiple Sequence Alignment

The problem of computing Multiple Sequence Alignments comes from bioinformatics, where it is typically used to find corresponding elements in proteins or DNA (Durbin et al., 1998).

A sequence alignment algorithm takes as its input some *sequences* $s_1, \ldots, s_n \in \Sigma^*$ over some alphabet Σ, along with a *cost function* $c_m : \Sigma \times \Sigma \to \mathbb{R}$ for substitutions and *gap costs* $c_{gap} \in \mathbb{R}$ for insertions and deletions. In bioinformatics, the elements of Σ could be nucleotides and a sequence could be a DNA sequence; in our case, Σ contains the individual event descriptions in our data, and the sequences are the ESDs.

A *Multiple Sequence Alignment* A of these sequences is then a matrix as in Fig. 2: The i-th column of A is the sequence s_i, possibly with some gaps ("⊘") interspersed between the symbols of s_i, such that each row contains at least one non-gap. If a row contains two non-gaps, we take these symbols to be *aligned*; aligning a non-gap with a gap can be thought of as an insertion or deletion.

Each sequence alignment A can be assigned a *cost* $c(A)$ in the following way:

$$c(A) = c_{gap} \cdot \Sigma_{\oslash} + \sum_{i=1}^{n} \sum_{\substack{j=1,\\ a_{ji} \neq \oslash}}^{m} \sum_{\substack{k=j+1,\\ a_{ki} \neq \oslash}}^{m} c_m(a_{ji}, a_{ki})$$

where $\Sigma_{\oslash}$ is the number of gaps in A, n is the number of rows and m the number of sequences. In other words, we sum up the alignment cost for any two symbols from Σ that are aligned with each other, and add the gap cost for each gap. There is an algorithm that computes cheapest pairwise alignments (i.e. $n = 2$) in polynomial time (Needleman and Wunsch, 1970). For $n > 2$, the problem is NP-complete, but there are efficient algorithms that approximate the cheapest MSAs by aligning two sequences first, considering the result as a single sequence whose elements are pairs, and repeating this process until all sequences are incorporated in the MSA (Higgins and Sharp, 1988).

5.2 Semantic similarity

In order to apply MSA to the problem of aligning ESDs, we choose Σ to be the set of all individual event descriptions in a given scenario. Intuitively, we want the MSA to prefer the alignment of two phrases if they are semantically similar, i.e. it should cost more to align 'exit' with 'eat' than 'exit' with 'leave'. Thus we take a measure of semantic (dis)similarity as the cost function c_m.

The phrases to be compared are written in bullet-point style. They are typically short and elliptic (no overt subject), they lack determiners and use infinitive or present progressive form for the main verb. Also, the lexicon differs considerably from usual newspaper corpora. For these reasons, standard methods for similarity assessment are not straightforwardly applicable: Simple bag-of-words approaches do not provide sufficiently good results, and standard taggers and parsers cannot process our descriptions with sufficient accuracy.

We therefore employ a simple, robust heuristics, which is tailored to our data and provides very

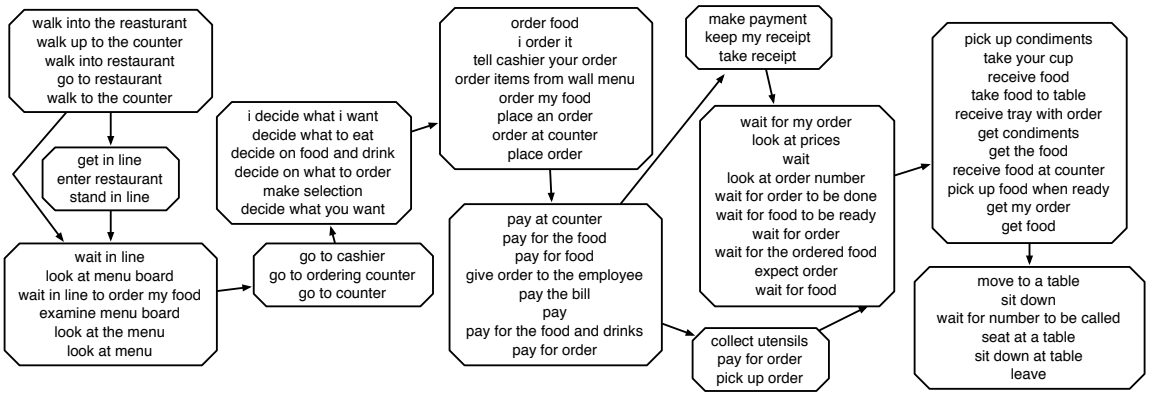


Figure 3: An extract from the graph computed for EATING IN A FAST FOOD RESTAURANT

shallow dependency-style syntactic information. We identify the first potential verb of the phrase (according to the POS information provided by WordNet) as the predicate, the preceding noun (if any) as subject, and all following potential nouns as objects. (With this fairly crude tagging method, we also count nouns in prepositional phrases as "objects".)

On the basis of this pseudo-parse, we compute the similarity measure sim:

$$sim = \alpha \cdot pred + \beta \cdot subj + \gamma \cdot obj$$

where $pred$, $subj$, and obj are the similarity values for predicates, subjects and objects respectively, and α, β, γ are weights. If a constituent is not present in one of the phrases to compare, we set its weight to zero and redistribute it over the other weights. We fix the individual similarity scores $pred$, $subj$, and obj depending on the WordNet relation between the most similar WordNet senses of the respective lemmas (100 for synonyms, 0 for lemmas without any relation, and intermediate numbers for different kind of WordNet links).

We optimized the values for $pred$, $subj$, and obj as well as the weights α, β and γ using a held-out development set of scenarios. Our experiments showed that in most cases, the verb contributes the largest part to the similarity (accordingly, α needs to be higher than the other factors). We achieved improved accuracy by distinguishing a class of verbs that contribute little to the meaning of the phrase (i.e., support verbs, verbs of movement, and the verb "get"), and assigning them a separate, lower α.

5.3 Building Temporal Script Graphs

We can now compute a low-cost MSA for each scenario out of the ESDs. From this alignment, we extract a temporal script graph, in the following way. First, we construct an initial graph which has one node for each row of the MSA as in Fig. 2. We interpret each node of the graph as representing a single event in the script, and the phrases that are collected in the node as different descriptions of this event; that is, we claim that these phrases are paraphrases in the context of this scenario. We then add an edge (u, v) to the graph iff (1) $u \neq v$, (2) there was at least one ESD in the original data in which some phrase in u directly preceded some phrase in v, and (3) if a single ESD contains a phrase from u and from v, the phrase from u directly precedes the one from v. In terms of the MSA, this means that if a phrase from u comes from the same column as a phrase from v, there are at most some gaps between them. This initial graph represents exactly the same information as the MSA, in a different notation.

The graph is automatically post-processed in a second step to simplify it and eliminate noise that caused MSA errors. At first we prune spurious nodes which contain only one event description. Then we refine the graph by merging nodes whose elements should have been aligned in the first place but were missed by the MSA. We merge two nodes if they satisfy certain structural and semantic constraints.

The *semantic* constraints check whether the event descriptions of the merged node would be sufficiently consistent according to the similarity measure from Section 5.2. To check whether we can merge two nodes u and v, we use an unsupervised clustering algorithm (Flake et al., 2004) to

first cluster the event descriptions in u and v separately. Then we combine the event descriptions from u and v and cluster the resulting set. If the union has more clusters than either u or v, we assume the nodes to be too dissimilar for merging.

The *structural* constraints depend on the graph structure. We only merge two nodes u and v if their event descriptions come from different sequences and one of the following conditions holds:

- u and v have the same parent;
- u has only one parent, v is its only child;
- v has only one child and is the only child of u;
- all children of u (except for v) are also children of v.

These structural constraints prevent the merging algorithm from introducing new temporal relations that are not supported by the input ESDs.

We take the output of this post-processing step as the temporal script graph. An excerpt of the graph we obtain for our running example is shown in Fig. 3. One node created by the node merging step was the top left one, which combines one original node containing 'walk into restaurant' and another with 'go to restaurant'. The graph mostly groups phrases together into event nodes quite well, although there are some exceptions, such as the 'collect utensils' node. Similarly, the temporal information in the graph is pretty accurate. But perhaps most importantly, our MSA-based algorithm manages to keep similar phrases like 'wait in line' and 'wait for my order' apart by exploiting the sequential structure of the input ESDs.

6 Evaluation

We evaluated the two core aspects of our system: its ability to recognize descriptions of the same event (*paraphrases*) and the resulting temporal constraints it defines on the event descriptions (*happens-before relation*). We compare our approach to two baseline systems and show that our system outperforms both baselines and sometimes even comes close to our upper bound.

6.1 Method

We selected ten scenarios which we did not use for development purposes, five of them taken from the corpus described in Section 4, the other five from the OMICS corpus.[2] The OMICS corpus is a freely available, web-collected corpus by the Open Mind Initiative (Singh et al., 2002). It contains several *stories* ($\approx$ scenarios) consisting of multiple ESDs. The corpus strongly resembles ours in language style and information provided, but is restricted to "indoor activities" and contains much more data than our collection (175 scenarios with more than 40 ESDs each).

For each scenario, we created a *paraphrase set* out of 30 randomly selected pairs of event descriptions which the system classified as paraphrases and 30 completely random pairs. The *happens-before set* consisted of 30 pairs classified as *happens-before*, 30 random pairs and additionally all 60 pairs in reverse order. We added the reversed pairs to check whether the raters really prefer one direction or whether they accept both and were biased by the order of presentation.

We presented each pair to 5 non-experts, all US residents, via Mechanical Turk. For the paraphrase set, an exemplary question we asked the rater looks as follows, instantiating the Scenario and the two descriptions to compare appropriately:

> Imagine two people, both telling a story about SCENARIO. Could the first one say *event*$_2$ to describe the same part of the story that the second one describes with *event*$_1$?

For the happens-before task, the question template was the following:

> Imagine somebody telling a story about SCENARIO in which the events *event*$_1$ and *event*$_2$ occur. Would *event*$_1$ normally happen before *event*$_2$?

We constructed a gold standard by a majority decision of the raters. An expert rater adjudicated the pairs with a 3:2 vote ratio.

6.2 Upper Bound and Baselines

To show the contributions of the different system components, we implemented two baselines:

Clustering Baseline: We employed an unsupervised clustering algorithm (Flake et al., 2004) and fed it all event descriptions of a scenario. We first created a similarity graph with one node per event description. Each pair of nodes is connected

[2] http://openmind.hri-us.com/

	Scenario	Precision			Recall			F-Score			
		sys	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	upper
MTurk	pay with credit card	0.52	0.43	0.50	0.84	0.89	0.11	**0.64**	0.58	• 0.17	0.60
	eat in restaurant	0.70	0.42	0.75	0.88	1.00	0.25	**0.78**	• 0.59	• 0.38	• 0.92
	iron clothes I	0.52	0.32	1.00	0.94	1.00	0.12	**0.67**	• 0.48	• 0.21	• 0.82
	cook scrambled eggs	0.58	0.34	0.50	0.86	0.95	0.10	**0.69**	• 0.50	• 0.16	• 0.91
	take a bus	0.65	0.42	0.40	0.87	1.00	0.09	**0.74**	• 0.59	• 0.14	• 0.88
OMICS	answer the phone	0.93	0.45	0.70	0.85	1.00	0.21	**0.89**	• 0.71	• 0.33	0.79
	buy from vending machine	0.59	0.43	0.59	0.83	1.00	0.54	**0.69**	0.60	0.57	0.80
	iron clothes II	0.57	0.30	0.33	0.94	1.00	0.22	**0.71**	• 0.46	• 0.27	0.77
	make coffee	0.50	0.27	0.56	0.94	1.00	0.31	**0.65**	• 0.42	○ 0.40	• 0.82
	make omelette	0.75	0.54	0.67	0.92	0.96	0.23	**0.83**	• 0.69	• 0.34	0.85
	Average	0.63	0.40	0.60	0.89	0.98	0.22	**0.73**	0.56	0.30	0.82

Figure 4: Results for paraphrasing task; significance of difference to *sys*: • : $p \leq 0.01$, ○ : $p \leq 0.1$

with a weighted edge; the weight reflects the semantic similarity of the nodes' event descriptions as described in Section 5.2. To include all input information on inequality of events, we did not allow for edges between nodes containing two descriptions occurring together in one ESD. The underlying assumption here is that two different event descriptions of the same ESD always represent distinct events.

The clustering algorithm uses a parameter which influences the cluster granularity, without determining the exact number of clusters beforehand. We optimized this parameter automatically for each scenario: The system picks the value that yields the optimal result with respect to density and distance of the clusters (Flake et al., 2004), i.e. the elements of each cluster are as similar as possible to each other, and as dissimilar as possible to the elements of all other clusters.

The clustering baseline considers two phrases as paraphrases if they are in the same cluster. It claims a happens-before relation between phrases e and f if some phrase in e's cluster precedes some phrase in f's cluster in the original ESDs. With this baseline, we can show the contribution of MSA.

Levenshtein Baseline: This system follows the same steps as our system, but using Levenshtein distance as the measure of semantic similarity for MSA and for node merging (cf. Section 5.3). This lets us measure the contribution of the more fine-grained similarity function. We computed Levenshtein distance as the character-wise edit distance on the phrases, divided by the phrases' character length so as to get comparable values for shorter and longer phrases. The gap costs for MSA with Levenshtein were optimized on our development set so as to produce the best possible alignment.

Upper bound: We also compared our system to a human-performance upper bound. Because no single annotator rated all pairs of ESDs, we constructed a "virtual annotator" as a point of comparison, by randomly selecting one of the human annotations for each pair.

6.3 Results

We calculated precision, recall, and f-score for our system, the baselines, and the upper bound as follows, with all_{system} being the number of pairs labelled as *paraphrase* or *happens-before*, all_{gold} as the respective number of pairs in the gold standard and $correct$ as the number of pairs labeled correctly by the system.

$$precision = \frac{correct}{all_{system}} \qquad recall = \frac{correct}{all_{gold}}$$

$$f\text{-}score = \frac{2 * precision * recall}{precision + recall}$$

The tables in Fig. 4 and 5 show the results of our system and the reference values; Fig. 4 describes the paraphrasing task and Fig. 5 the happens-before task. The upper half of the tables describes the test sets from our own corpus, the remainder refers to OMICS data. The columns labelled *sys* contain the results of our system, *base$_{cl}$* describes the clustering baseline and *base$_{lev}$* the Levenshtein baseline. The f-score for the upper bound is in the column *upper*. For the f-score values, we calculated the significance for the difference between our system and the baselines as well as the upper bound, using a resampling test (Edgington, 1986). The values marked with • differ from our system significantly at a level of $p \leq 0.01$, ○ marks a level of $p \leq 0.1$. The remaining values are not significant with $p \leq 0.1$. (For the average values, no sig-

	Scenario	Precision			Recall			F-Score			
		sys	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	upper
MTurk	pay with credit card	0.86	0.49	0.65	0.84	0.74	0.45	**0.85**	• 0.59	• 0.53	0.92
	eat in restaurant	0.78	0.48	0.68	0.84	0.98	0.75	**0.81**	• 0.64	0.71	• 0.95
	iron clothes I	0.78	0.54	0.75	0.72	0.95	0.53	**0.75**	0.69	• 0.62	• 0.92
	cook scrambled eggs	0.67	0.54	0.55	0.64	0.98	0.69	0.66	***0.70***	0.61	• 0.88
	take a bus	0.80	0.49	0.68	0.80	1.00	0.37	**0.80**	• 0.66	• 0.48	• 0.96
OMICS	answer the phone	0.83	0.48	0.79	0.86	1.00	0.96	0.84	• 0.64	***0.87***	0.90
	buy from vending machine	0.84	0.51	0.69	0.85	0.90	0.75	**0.84**	• 0.66	○ 0.71	0.83
	iron clothes II	0.78	0.48	0.75	0.80	0.96	0.66	**0.79**	• 0.64	0.70	0.84
	make coffee	0.70	0.55	0.50	0.78	1.00	0.55	**0.74**	0.71	○ 0.53	○ 0.83
	make omelette	0.70	0.55	0.79	0.83	0.93	0.82	0.76	○ 0.69	***0.81***	• 0.92
	Average	0.77	0.51	0.68	0.80	0.95	0.65	**0.78**	0.66	0.66	0.90

Figure 5: Results for happens-before task; significance of difference to *sys*: • : $p \leq 0.01$, ○ : $p \leq 0.1$

nificance is calculated because this does not make sense for scenario-wise evaluation.)

Paraphrase task: Our system outperforms both baselines clearly, reaching significantly higher f-scores in 17 of 20 cases. Moreover, for five scenarios, the upper bound does not differ significantly from our system. For judging the precision, consider that the test set is slightly biased: Labeling all pairs with the majority category (*no paraphrase*) would result in a precision of 0.64. However, recall and f-score for this trivial lower bound would be 0.

The only scenario in which our system doesn't score very well is BUY FROM A VENDING MACHINE, where the upper bound is not significantly better either. The clustering system, which can't exploit the sequential information from the ESDs, has trouble distinguishing semantically similar phrases (high recall, low precision). The Levenshtein similarity measure, on the other hand, is too restrictive and thus results in comparatively high precisions, but very low recall.

Happens-before task: In most cases, and on average, our system is superior to both baselines. Where a baseline system performs better than ours, the differences are not significant. In four cases, our system does not differ significantly from the upper bound. Regarding precision, our system outperforms both baselines in all scenarios except one (MAKE OMELETTE).

Again the clustering baseline is not fine-grained enough and suffers from poor precision, only slightly better than the majority baseline. The Levenshtein baseline gets mostly poor recall, except for ANSWER THE PHONE: to describe this scenario, people used very similar wording. In such a scenario, adding lexical knowledge to the sequential information makes less of a difference.

On average, the baselines do much better here than for the paraphrase task. This is because once a system decides on paraphrase clusters that are essentially correct, it can retrieve correct information about the temporal order directly from the original ESDs.

Both tables illustrate that the task complexity strongly depends on the scenario: Scripts that allow for a lot of variation with respect to ordering (such as COOK SCRAMBLED EGGS) are particularly challenging for our system. This is due to the fact that our current system can neither represent nor find out that two events can happen in arbitrary order (e.g., 'take out pan' and 'take out bowl').

One striking difference between the performance of our system on the OMICS data and on our own dataset is the relation to the upper bound: On our own data, the upper bound is almost always significantly better than our system, whereas significant differences are rare on OMICS. This difference bears further analysis; we speculate it might be caused either by the increased amount of training data in OMICS or by differences in language (e.g., fewer anaphoric references).

7 Conclusion

We conclude with a summary of this paper and some discussion along with hints to future work in the last part.

7.1 Summary

In this paper, we have described a novel approach to the unsupervised learning of temporal script information. Our approach differs from previous work in that we collect training data by directly asking non-expert users to describe a scenario, and

then apply a Multiple Sequence Alignment algorithm to extract scenario-specific paraphrase and temporal ordering information. We showed that our system outperforms two baselines and sometimes approaches human-level performance, especially because it can exploit the sequential structure of the script descriptions to separate clusters of semantically similar events.

7.2 Discussion and Future Work

We believe that we can scale this approach to model a large numbers of scenarios representing implicit shared knowledge. To realize this goal, we are going to automatize several processing steps that were done manually for the current study. We will restrict the user input to lexicon words to avoid manual orthography correction. Further, we will implement some heuristics to filter unusable instances by matching them with the remaining data. As far as the data collection is concerned, we plan to replace the web form with a browser game, following the example of von Ahn and Dabbish (2008). This game will feature an algorithm that can generate new candidate scenarios without any supervision, for instance by identifying suitable sub-events of collected scripts (e.g. inducing data collection for PAY as *sub*-event sequence of GO SHOPPING)

On the technical side, we intend to address the question of detecting participants of the scripts and integrating them into the graphs, Further, we plan to move on to more elaborate data structures than our current TSGs, and then identify and represent script elements like optional events, alternative events for the same step, and events that can occur in arbitrary order.

Because our approach gathers information from volunteers on the Web, it is limited by the knowledge of these volunteers. We expect it will perform best for general commonsense knowledge; culture-specific knowledge or domain-specific expert knowledge will be hard for it to learn. This limitation could be addressed by targeting specific groups of online users, or by complementing our approach with corpus-based methods, which might perform well exactly where ours does not.

Acknowledgements

We want to thank Dustin Smith for the OMICS data, Alexis Palmer for her support with Amazon Mechanical Turk, Nils Bendfeldt for the creation of all web forms and Ines Rehbein for her effort with several parsing experiments. In particular, we thank the anonymous reviewers for their helpful comments. – This work was funded by the Cluster of Excellence "Multimodal Computing and Interaction" in the German Excellence Initiative.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA. Association for Computational Linguistics.

Avron Barr and Edward Feigenbaum. 1981. *The Handbook of Artificial Intelligence, Volume 1*. William Kaufman Inc., Los Altos, CA.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL 2003*.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2009. A demonstration of human computation using the phrase detectives annotation game. In *KDD Workshop on Human Computation*. ACM.

Nathanael Chambers and Dan Jurafsky. 2008a. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of EMNLP 2008*.

Nathanael Chambers and Dan Jurafsky. 2008b. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-IJCNLP 2009*.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of ACL-07: Interactive Poster and Demonstration Sessions*.

Richard Edward Cullingford. 1977. *Script application: computer understanding of newspaper stories.* Ph.D. thesis, Yale University, New Haven, CT, USA.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis*. Cambridge University Press.

Eugene S Edgington. 1986. *Randomization tests*. Marcel Dekker, Inc., New York, NY, USA.

Gary W. Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. 2004. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4).

Andrew S. Gordon. 2001. Browsing image collections with representations of common-sense activities. *JASIST*, 52(11).

Desmond G. Higgins and Paul M. Sharp. 1988. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1).

Dominic R. Jones and Cynthia A. Thompson. 2003. Identifying events using similarity and context. In *Proceedings of CoNNL-2003*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *COLING/ACL-2006*.

Mehdi Manshadi, Reid Swanson, and Andrew S. Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the 21st FLAIRS Conference*.

Michael McTear. 1987. *The Articulate Computer*. Blackwell Publishers, Inc., Cambridge, MA, USA.

Risto Miikkulainen. 1995. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence*, 5(2), 04.

Raymond J. Mooney. 1990. Learning plan schemata from observation: Explanation-based learning for plan recognition. *Cognitive Science*, 14(4).

Erik T. Mueller. 1998. *Natural Language Processing with Thought Treasure*. Signiform.

Erik T. Mueller. 2004. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research*, 5(4).

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3), March.

Lisa F. Rau, Paul S. Jacobs, and Uri Zernik. 1989. Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing and Management*, 25(4):419 – 428.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, Hillsdale, NJ.

Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan L. Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems - DOA, CoopIS and ODBASE 2002*, London, UK. Springer-Verlag.

Dustin Smith and Kenneth C. Arnold. 2009. Learning hierarchical plans by reading simple english narratives. In *Proceedings of the Commonsense Workshop at IUI-09*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP 2008*.

Reid Swanson and Andrew S. Gordon. 2008. Say anything: A massively collaborative open domain story writing companion. In *Proceedings of ICIDS 2008*.

Luis von Ahn and Laura Dabbish. 2008. Designing games with a purpose. *Commun. ACM*, 51(8).

Starting From Scratch in Semantic Role Labeling

Michael Connor
University of Illinois
connor2@uiuc.edu

Yael Gertner
University of Illinois
ygertner@cyrus.psych.uiuc.edu

Cynthia Fisher
University of Illinois
cfisher@cyrus.psych.uiuc.edu

Dan Roth
University of Illinois
danr@illinois.edu

Abstract

A fundamental step in sentence comprehension involves assigning semantic roles to sentence constituents. To accomplish this, the listener must parse the sentence, find constituents that are candidate arguments, and assign semantic roles to those constituents. Each step depends on prior lexical and syntactic knowledge. Where do children learning their first languages begin in solving this problem? In this paper we focus on the parsing and argument-identification steps that precede Semantic Role Labeling (SRL) training. We combine a simplified SRL with an unsupervised HMM part of speech tagger, and experiment with psycholinguistically-motivated ways to label clusters resulting from the HMM so that they can be used to parse input for the SRL system. The results show that proposed shallow representations of sentence structure are robust to reductions in parsing accuracy, and that the contribution of alternative representations of sentence structure to successful semantic role labeling varies with the integrity of the parsing and argument-identification stages.

1 Introduction

In this paper we present experiments with an automatic system for semantic role labeling (SRL) that is designed to model aspects of human language acquisition. This simplified SRL system is inspired by the syntactic bootstrapping theory, and by an account of syntactic bootstrapping known as 'structure-mapping' (Fisher, 1996; Gillette et al., 1999; Lidz et al., 2003). Syntactic bootstrapping theory proposes that young children use their very partial knowledge of syntax to guide sentence comprehension. The structure-mapping account makes three key assumptions: First, sentence comprehension is grounded by the acquisition of an initial set of concrete nouns. Nouns are arguably less dependent on prior linguistic knowledge for their acquisition than are verbs; thus children are assumed to be able to identify the referents of some nouns via cross-situational observation (Gillette et al., 1999). Second, these nouns, once identified, yield a skeletal sentence structure. Children treat each noun as a candidate argument, and thus interpret the number of nouns in the sentence as a cue to its semantic predicate-argument structure (Fisher, 1996). Third, children represent sentences in an abstract format that permits generalization to new verbs (Gertner et al., 2006).

The structure-mapping account of early syntactic bootstrapping makes strong predictions, including predictions of tell-tale errors. In the sentence "Ellen and John laughed", an intransitive verb appears with two nouns. If young children rely on representations of sentences as simple as an ordered set of nouns, then they should have trouble distinguishing such sentences from transitive sentences. Experimental evidence suggests that they do: 21-month-olds mistakenly interpreted word order in sentences such as "The girl and the boy kradded" as conveying agent-patient roles (Gertner and Fisher, 2006).

Previous computational experiments with a system for automatic semantic role labeling (BabySRL: (Connor et al., 2008)) showed that it is possible to learn to assign basic semantic roles based on the shallow sentence representations proposed by the structure-mapping view. Furthermore, these simple structural features were robust to drastic reductions in the integrity of the semantic-role feedback (Connor et al., 2009). These experiments showed that representations of sentence structure as simple as 'first of two nouns' are useful, but the experiments relied on perfect

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 989–998,
Uppsala, Sweden, 11-16 July 2010.

knowledge of arguments and predicates as a start to classification.

Perfect built-in parsing finesses two problems facing the human learner. The first problem involves classifying words by part-of-speech. Proposed solutions to this problem in the NLP and human language acquisition literatures focus on distributional learning as a key data source (e.g., (Mintz, 2003; Johnson, 2007)). Importantly, infants are good at learning distributional patterns (Gomez and Gerken, 1999; Saffran et al., 1996). Here we use a fairly standard Hidden Markov Model (HMM) to generate clusters of words that occur in similar distributional contexts in a corpus of input sentences.

The second problem facing the learner is more contentious: Having identified clusters of distributionally-similar words, how do children figure out what role these clusters of words should play in a sentence interpretation system? Some clusters contain nouns, which are candidate arguments; others contain verbs, which take arguments. How is the child to know which are which? In order to use the output of the HMM tagger to process sentences for input to an SRL model, we must find a way to automatically label the clusters.

Our strategies for automatic argument and predicate identification, spelled out below, reflect core claims of the structure-mapping theory: (1) The meanings of some concrete nouns can be learned without prior linguistic knowledge; these concrete nouns are assumed based on their meanings to be possible arguments; (2) verbs are identified, not primarily by learning their meanings via observation, but rather by learning about their syntactic argument-taking behavior in sentences.

By using the HMM part-of-speech tagger in this way, we can ask how the simple structural features that we propose children start with stand up to reductions in parsing accuracy. In doing so, we move to a parser derived from a particular theoretical account of how the human learner might classify words, and link them into a system for sentence comprehension.

2 Model

We model language learning as a Semantic Role Labeling (SRL) task (Carreras and Màrquez, 2004). This allows us to ask whether a learner, equipped with particular theoretically-motivated representations of the input, can learn to understand sentences at the level of who did what to whom. The architecture of our system is similar to a previous approach to modeling early language acquisition (Connor et al., 2009), which is itself based on the standard architecture of a full SRL system (e.g. (Punyakanok et al., 2008)).

This basic approach follows a multi-stage pipeline, with each stage feeding in to the next. The stages are: (1) Parsing the sentence, (2) Identifying potential predicates and arguments based on the parse, (3) Classifying role labels for each potential argument relative to a predicate, (4) Applying constraints to find the best labeling of arguments for a sentence. In this work we attempt to limit the knowledge available at each stage to the automatic output of the previous stage, constrained by knowledge that we argue is available to children in the early stages of language learning.

In the parsing stage we use an unsupervised parser based on Hidden Markov Models (HMM), modeling a simple 'predict the next word' parser. Next the argument identification stage identifies HMM states that correspond to possible arguments and predicates. The candidate arguments and predicates identified in each input sentence are passed to an SRL classifier that uses simple abstract features based on the number and order of arguments to learn to assign semantic roles.

As input to our learner we use samples of natural child directed speech (CDS) from the CHILDES corpora (MacWhinney, 2000). During initial unsupervised parsing we experiment with incorporating knowledge through a combination of statistical priors favoring a skewed distribution of words into classes, and an initial hard clustering of the vocabulary into function and content words. The argument identifier uses a small set of frequent nouns to seed argument states, relying on the assumptions that some concrete nouns can be learned as a prerequisite to sentence interpretation, and are interpreted as candidate arguments.

The SRL classifier starts with noisy largely unsupervised argument identification, and receives feedback based on annotation in the PropBank style; in training, each word identified as an argument receives the true role label of the phrase that word is part of. This represents the assumption that learning to interpret sentences is naturally supervised by the fit of the learner's predicted meaning with the referential context. The provision

of perfect 'gold-standard' feedback over-estimates the real child's access to this supervision, but allows us to investigate the consequences of noisy argument identification for SRL performance. We show that even with imperfect parsing, a learner can identify useful abstract patterns for sentence interpretation. Our ultimate goal is to 'close the loop' of this system, by using learning in the SRL system to improve the initial unsupervised parse and argument identification.

The training data were samples of parental speech to three children (Adam, Eve, and Sarah; (Brown, 1973)), available via CHILDES. The SRL training corpus consists of parental utterances in samples Adam 01-20 (child age 2;3 - 3;1), Eve 01-18 (1;6 - 2;2), and Sarah 01-83 (2;3 - 3;11). All verb-containing utterances without symbols indicating disfluencies were automatically parsed with the Charniak parser (Charniak, 1997), annotated using an existing SRL system (Punyakanok et al., 2008) and then errors were hand-corrected. The final annotated sample contains about 16,730 propositions, with 32,205 arguments.

3 Unsupervised Parsing

As a first step of processing, we feed the learner large amounts of unlabeled text and expect it to learn some structure over this data that will facilitate future processing. The source of this text is child directed speech collected from various projects in the CHILDES repository[1]. We removed sentences with fewer than three words or markers of disfluency. In the end we used 160 thousand sentences from this set, totaling over 1 million tokens and 10 thousand unique words.

The goal of the parsing stage is to give the learner a representation permitting it to generalize over word forms. The exact parse we are after is a distributional and context-sensitive clustering of words based on sequential processing. We chose an HMM based parser for this since, in essence the HMM yields an unsupervised POS classifier, but without names for states. An HMM trained with expectation maximization (EM) is analogous to a simple process of predicting the next word in a stream and correcting connections accordingly for each sentence.

With HMM we can also easily incorporate additional knowledge during parameter estimation. The first (and simplest) parser we used was an HMM trained using EM with 80 hidden states. The number of hidden states was made relatively large to increase the likelihood of clusters corresponding to a single part of speech, while preserving some degree of generalization.

Johnson (2007) observed that EM tends to create word clusters of uniform size, which does not reflect the way words cluster into parts of speech in natural languages. The addition of priors biasing the system toward a skewed allocation of words to classes can help. The second parser was an 80 state HMM trained with Variational Bayes EM (VB) incorporating Dirichlet priors (Beal, 2003).[2]

In the third and fourth parsers we experiment with enriching the HMM POS-tagger with other psycholinguistically plausible knowledge. Words of different grammatical categories differ in their phonological as well as in their distributional properties (e.g., (Kelly, 1992; Monaghan et al., 2005; Shi et al., 1998)); combining phonological and distributional information improves the clustering of words into grammatical categories. The phonological difference between content and function words is particularly striking (Shi et al., 1998). Even newborns can categorically distinguish content and function words, based on the phonological difference between the two classes (Shi et al., 1999). Human learners may treat content and function words as distinct classes from the start.

To implement this division into function and content words[3], we start with a list of function word POS tags[4] and then find words that appear predominantly with these POS tags, using tagged WSJ data (Marcus et al., 1993). We allocated a fixed number of states for these function words, and left the rest of the states for the rest of the words. This amounts to initializing the emission matrix for the HMM with a block structure; words from one class cannot be emitted by states allocated to the other class. This trick has been used before in speech recognition work (Rabiner,

[1] We used parts of the Bloom (Bloom, 1970; Bloom, 1973), Brent (Brent and Siskind, 2001), Brown (Brown, 1973), Clark (Clark, 1978), Cornell, MacWhinney (MacWhinney, 2000), Post (Demetras et al., 1986) and Providence (Demuth et al., 2006) collections.

[2] We tuned the prior using the same set of 8 value pairs suggested by Gao and Johnson (2008), using a held out set of POS-tagged CDS to evaluate final performance.

[3] We also include a small third class for punctuation, which is discarded.

[4] TO,IN,EX,POS,WDT,PDT,WRB,MD,CC,DT,RP,UH

1989), and requires far fewer resources than the full tagging dictionary that is often used to intelligently initialize an unsupervised POS classifier (e.g. (Brill, 1997; Toutanova and Johnson, 2007; Ravi and Knight, 2009)).

Because the function and content word preclustering preceded parameter estimation, it can be combined with either EM or VB learning. Although this initial split forces sparsity on the emission matrix and allows more uniform sized clusters, Dirichlet priors may still help, if word clusters within the function or content word subsets vary in size and frequency. The third parser was an 80 state HMM trained with EM estimation, with 30 states pre-allocated to function words; the fourth parser was the same except that it was trained with VB EM.

3.1 Parser Evaluation

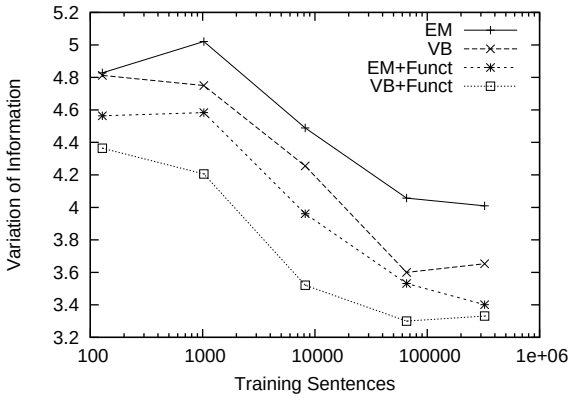


Figure 1: Unsupervised Part of Speech results, matching states to gold POS labels. All systems use 80 states, and comparison is to gold labeled CDS text, which makes up a subset of the HMM training data. Variation of Information is an information-theoretic measure summing mutual information between tags and states, proposed by (Meilă, 2002), and first used for Unsupervised Part of Speech in (Goldwater and Griffiths, 2007). Smaller numbers are better, indicating less information lost in moving from the HMM states to the gold POS tags. Note that incorporating function word precluster-ing allows both EM and VB algorithms to achieve the same performance with an order of magnitude fewer sentences.

We first evaluate these parsers (the first stage of our SRL system) on unsupervised POS tagging. Figure 1 shows the performance of the four systems using Variation of Information to measure match between gold states and unsupervised parsers as we vary the amount of text they receive. Each point on the graph represents the average result over 10 runs of the HMM with different samples of the unlabeled CDS. Another common measure for unsupervised POS (when there are more states than tags) is a many to one greedy mapping of states to tags. It is known that EM gives a better many to one score than VB trained HMM (Johnson, 2007), and likewise we see that here: with all data EM gives 0.75 matching, VB gives 0.74, while both EM+Funct and VB+Funct reach 0.80.

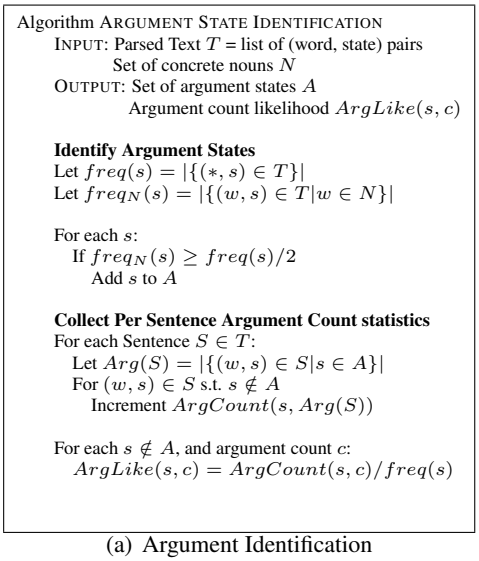
Adding the function/content word split to the HMM structure improves both EM and VB estimation in terms of both tag matching accuracy and information. However, these measures look at the parser only in isolation. What is more important to us is how useful the provided word clusters are for future semantic processing. In the next sections we use the outputs of our four parsers to identify arguments and predicates.

4 Argument Identification

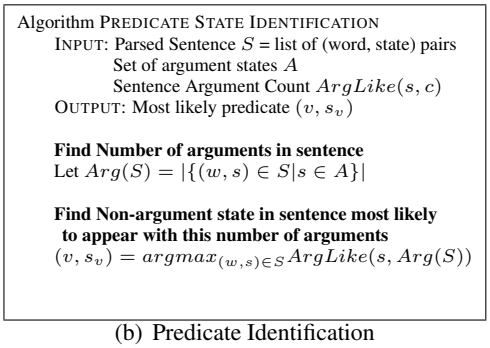
The unsupervised parser provides a state label for each word in each sentence; the goal of the argument identification stage is to use these states to label words as potential arguments, predicates or neither. As described in the introduction, core premises of the structure-mapping account offer routes whereby we could label some HMM states as argument or predicate states.

The structure-mapping account holds that sentence comprehension is grounded in the learning of an initial set of nouns. Children are assumed to identify the referents of some concrete nouns via cross-situational learning (Gillette et al., 1999; Smith and Yu, 2008). Children then assume, by virtue of the meanings of these nouns, that they are candidate arguments. This is a simple form of semantic bootstrapping, requiring the use of built-in links between semantics and syntax to identify the grammatical type of known words (Pinker, 1984). We use a small set of known nouns to transform unlabeled word clusters into candidate arguments for the SRL: HMM states that are dominated by known names for animate or inanimate objects are assumed to be argument states.

Given text parsed by the HMM parser and a list of known nouns, the argument identifier proceeds in multiple steps as illustrated in figure 2. The first stage identifies as argument states those states that appear at least half the time in the training data with known nouns. This use of a seed list and distributional clustering is similar to Prototype Driven Learning (Haghighi and Klein, 2006), except we are only providing information on one specific class.



(a) Argument Identification



(b) Predicate Identification

Figure 2: Argument identification algorithm. This is a two stage process: argument state identification based on statistics collected over entire text and per sentence predicate identification.

As a list of known nouns we collected all those nouns that appear three times or more in the child directed speech training data and judged to be either animate or inanimate nouns. The full set of 365 nouns covers over 93% of noun occurences in our data. In upcoming sections we experiment with varying the number of seed nouns used from this set, selecting the most frequent set of nouns. Reflecting the spoken nature of the child directed speech, the most frequent nouns are pronouns, but beyond the top 10 we see nouns naming people ('daddy', 'ursula') and object nouns ('chair', 'lunch').

What about verbs? A typical SRL model identifies candidate arguments and tries to assign roles to them relative to each verb in the sentence. In principle one might suppose that children learn the meanings of verbs via cross-situational observation just as they learn the meanings of concrete nouns. But identifying the meanings of verbs is much more troublesome. Verbs' meanings are abstract, therefore harder to identify based on scene information alone (Gillette et al., 1999). As a result, early vocabularies are dominated by nouns (Gentner, 2006). On the structure-mapping account, learners identify verbs, and begin to determine their meanings, based on sentence structure cues. Verbs take noun arguments; thus, learners could learn which words are verbs by detecting each verb's syntactic argument-taking behavior. Experimental evidence provides some support for this procedure: 2-year-olds keep track of the syntactic structures in which a new verb appears, even without a concurrent scene that provides cues to the verb's semantic content (Yuan and Fisher, 2009).

We implement this behavior by identifying as predicate states the HMM states that appear commonly with a particular number of previously identified arguments. First, we collect statistics over the entire HMM training corpus regarding how many arguments are identified per sentence, and which states that are not identified as argument states appear with each number of arguments. Next, for each parsed sentence that serves as SRL input, the algorithm chooses as the most likely predicate the word whose state is most likely to appear with the number of arguments found in the current input sentence. Note that this algorithm assumes exactly one predicate per sentence. Implicitly, the argument count likelihood divides predicate states up into transitive and intransitive predicates based on appearances in the simple sentences of CDS.

4.1 Argument Identification Evaluation

Figure 3 shows argument and predicate identification accuracy for each of the four parsers when provided with different numbers of known nouns. The known word list is very skewed with its most frequent members dominating the total noun occurrences in the data. The ten most frequent words[5] account for 60% of the total noun occurrences. We achieve the different occurrence coverage numbers of figure 3 by using the most frequent N words from the list that give the specific coverage[6]. Pronouns refer to people or objects, but are abstract in that they can refer to any person or object. The inclusion of pronouns in our list of

[5] you, it, I, what, he, me, ya, she, we, her

[6] N of 5, 10, 30, 83, 227 cover 50%, 60%, 70%, 80%, 90% of all noun occurrences

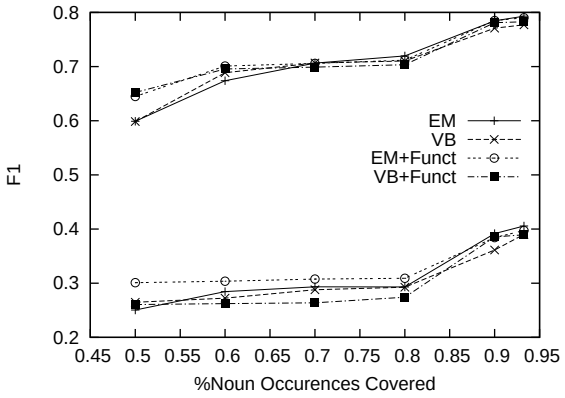


Figure 3: Effect of number of concrete nouns for seeding argument identification with various unsupervised parsers. Argument identification accuracy is computed against true argument boundaries from hand labeled data. The upper set of results show primary argument (A0-4) identification F1, and bottom lines show predicate identification F1.

known nouns represents the assumption that toddlers have already identified pronouns as referential terms. Even 19-month-olds assign appropriately different interpretations to novel verbs presented in simple transitive versus intransitive sentences with pronoun arguments ("He's kradding him!" vs. "He's kradding!"; (Yuan et al., 2007)). In ongoing work we experiment with other methods of identifying seed nouns.

Two groups of curves appear in figure 3: the upper group shows the primary argument identification accuracy and the bottom group shows the predicate identification accuracy. We evaluate compared to gold tagged data with true argument and predicate boundaries. The primary argument (A0-4) identification accuracy is the F1 value, with precision calculated as the proportion of identified arguments that appear as part of a true argument, and recall as the proportion of true arguments that have some state identified as an argument. F1 is calculated similarly for predicate identification, as one state per sentence is identified as the predicate.

As shown in figure 3, argument identification F1 is higher than predicate identification (which is to be expected, given that predicate identification depends on accurate arguments), and as we add more seed nouns the argument identification improves. Surprisingly, despite the clear differences in unsupervised POS performance seen in figure 1, the different parsers do not yield very different argument and predicate identification. As we will see in the next section, however, when the arguments identified in this step are used to train SRL classifier, distinctions between parsers reappear, suggesting that argument identification F1 masks systematic patterns in the errors.

5 Testing SRL Performance

Finally, we used the results of the previous parsing and argument-identification stages in training a simplified SRL classifier (BabySRL), equipped with sets of features derived from the structure-mapping account. For argument classification we used a linear classifier trained with a regularized perceptron update rule (Grove and Roth, 2001). In the results reported below the BabySRL did not use sentence-level inference for the final classification, every identified argument is classified independently; thus multiple nouns can have the same role. In what follows, we compare the performance of the BabySRL across the four parsers. We evaluated SRL performance by testing the BabySRL with constructed sentences like those used for the experiments with children described in the Introduction. All test sentences contained a novel verb, to test the model's ability to generalize.

We examine the performance of four versions of the BabySRL, varying in the features used to represent sentences. All four versions include lexical features consisting of the target argument and predicate (as identified in the previous steps). The baseline model has only these lexical features (Lexical). Following Connor et al. (2008; 2009), the key feature type we propose is noun pattern features (NounPat). Noun pattern features indicate how many nouns there are in the sentence and which noun the target is. For example, in "You dropped it!", 'you' has a feature active indicating that it is the first of two nouns, while 'it' has a feature active indicating that it is the second of two nouns. We compared the behavior of noun pattern features to another simple representation of word order, position relative to the verb (VerbPos). In the same example sentence, 'you' has a feature active indicating that it is pre-verbal; for 'it' a feature is active indicating that it is post-verbal. A fourth version of the BabySRL (Combined) used both NounPat and VerbPos features.

We structured our tests of the BabySRL to test the predictions of the structure-mapping account. (1) NounPat features will improve the SRL's ability to interpret simple transitive test sentences containing two nouns and a novel verb, relative

to a lexical baseline. Like 21-month-old children (Gertner et al., 2006), the SRL should interpret the first noun as an agent and the second as a patient. (2) Because NounPat features represent word order solely in terms of a sequence of nouns, an SRL equipped with these features will make the errors predicted by the structure-mapping account and documented in children (Gertner and Fisher, 2006). (3) NounPat features permit the SRL to assign different roles to the subjects of transitive and intransitive sentences that differ in their number of nouns. This effect follows from the nature of the NounPat features: These features partition the training data based on the number of nouns, and therefore learn separately the likely roles of the '1st of 1 noun' and the '1st of 2 nouns'.

These patterns contrast with the behavior of the VerbPos features: When the BabySRL was trained with perfect parsing, VerbPos promoted agent-patient interpretations of transitive test sentences, and did so even more successfully than NounPat features did, reflecting the usefulness of position relative to the verb in understanding English sentences. In addition, VerbPos features eliminated the errors with two-noun intransitive sentences. Given test sentences such as 'You and Mommy krad', VerbPos features represented both nouns as pre-verbal, and therefore identified both as likely agents. However, VerbPos features did not help the SRL assign different roles to the subjects of simple transitive and intransitive sentences: 'Mommy' in 'Mommy krads you' and 'Mommy krads' are both represented simply as pre-verbal.

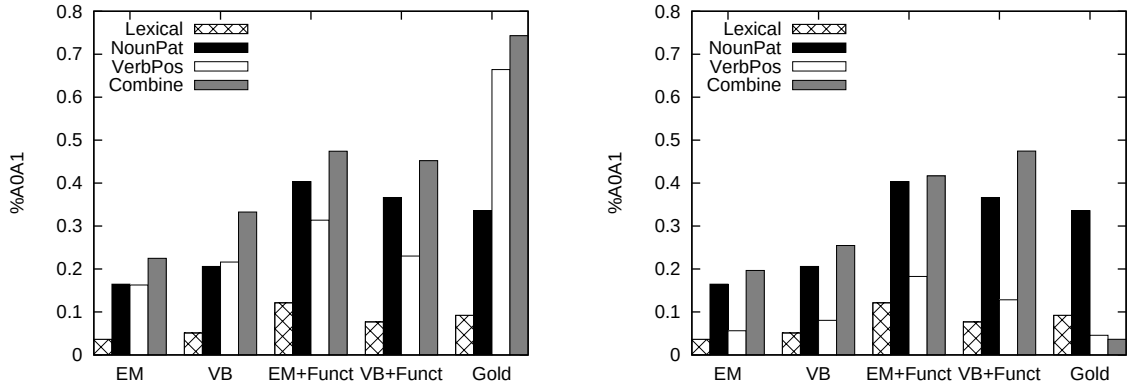
To test the system's predictions on transitive and intransitive two noun sentences, we constructed two test sentence templates: 'A krads B' and 'A and B krad', where A and B were replaced with familiar animate nouns. The animate nouns were selected from all three children's data in the training set and paired together in the templates such that all pairs are represented.

Figure 4 shows SRL performance on test sentences containing a novel verb and two animate nouns. Each plot shows the proportion of test sentences that were assigned an agent-patient (A0-A1) role sequence; this sequence is correct for transitive sentences but is an error for two-noun intransitive sentences. Each group of bars shows the performance of the BabySRL trained using one of the four parsers, equipped with each of our four feature sets. The top and bottom panels in Figure 4 differ in the number of nouns provided to seed the argument identification stage. The top row shows performance with 10 seed nouns (the 10 most frequent nouns, mostly animate pronouns), and the bottom row shows performance with 365 concrete (animate or inanimate) nouns treated as known. Relative to the lexical baseline, NounPat features fared well: they promoted the assignment of A0-A1 interpretations to transitive sentences, across all parser versions and both sets of known nouns. Both VB estimation and the content-function word split increased the ability of NounPat features to learn that the first of two nouns was an agent, and the second a patient. The NounPat features also promote the predicted error with two-noun intransitive sentences (Figures 4(b), 4(d)). Despite the relatively low accuracy of predicate identification noted in section 4.1, the VerbPos features did succeed in promoting an A0A1 interpretation for transitive sentences containing novel verbs relative to the lexical baseline. In every case the performance of the Combined model that includes both NounPat and VerbPos features exceeds the performance of either NounPat or VerbPos alone, suggesting both contribute to correct predictions for transitive sentences. However, the performance of VerbPos features did not improve with parsing accuracy as did the performance of the NounPat features. Most strikingly, the VerbPos features did not eliminate the predicted error with two-noun intransitive sentences, as shown in panels 4(b) and 4(d). The Combined model predicted an A0A1 sequence for these sentences, showing no reduction in this error due to the participation of VerbPos features.

Table 1 shows SRL performance on the same transitive test sentences ('A krads B'), compared to simple one-noun intransitive sentences ('A krads'). To permit a direct comparison, the table reports the proportion of transitive test sentences for which the first noun was assigned an agent (A0) interpretation, and the proportion of intransitive test sentences with the agent (A0) role assigned to the single noun in the sentence. Here we report only the results from the best-performing parser (trained with VB EM, and content/function word pre-clustering), compared to the same classifiers trained with gold standard argument identification. When trained on arguments identified via the unsupervised POS tagger, noun pattern features promoted agent interpretations of tran-

	Two Noun Transitive, % Agent First				One Noun Intransitive, % Agent Prediction			
	Lexical	NounPat	VerbPos	Combine	Lexical	NounPat	VerbPos	Combine
VB+Funct 10 seed	0.48	0.61	0.55	0.71	0.48	0.57	0.56	0.59
VB+Funct 365 seed	0.22	0.64	0.41	0.74	0.23	0.33	0.43	0.41
Gold Arguments	0.16	0.41	0.69	0.77	0.17	0.18	0.70	0.58

Table 1: SRL result comparison when trained with best unsupervised argument identifier versus trained with gold arguments. Comparison is between agent first prediction of two noun transitive sentences vs. one noun intransitive sentences. The unsupervised arguments lead the classifier to rely more on noun pattern features; when the true arguments and predicate are known the verb position feature leads the classifier to strongly indicate agent first in both settings.



(a) Two Noun Transitive Sentence, 10 seed nouns

(b) Two Noun Intransitive Sentence, 10 seed nouns

(c) Two Noun Transitive Sentence, 365 seed nouns

(d) Two Noun Intransitive Sentence, 365 seed nouns

Figure 4: SRL classification performance on transitive and intransitive test sentences containing two nouns and a novel verb. Performance with gold-standard argument identification is included for comparison. Across parses, noun pattern features promote agent-patient (A0A1) interpretations of both transitive ("You krad Mommy") and two-noun intransitive sentences ("You and Mommy krad"); the latter is an error found in young children. Unsupervised parsing is less accurate in identifying the verb, so verb position features fail to eliminate errors with two-noun intransitive sentences.

sitive subjects, but not for intransitive subjects. This differentiation between transitive and intransitive sentences was clearer when more known nouns were provided. Verb position features, in contrast, promote agent interpretations of subjects weakly with unsupervised argument identification, but equally for transitive and intransitive.

Noun pattern features were robust to increases in parsing noise. The behavior of verb position features suggests that variations in the identifiability of different parts of speech can affect the usefulness of alternative representations of sentence structure. Representations that reflect the position of the verb may be powerful guides for understanding simple English sentences, but representations reflecting only the number and order of nouns can dominate early in acquisition, depending on the integrity of parsing decisions.

6 Conclusion and Future Work

The key innovation in the present work is the combination of unsupervised part-of-speech tagging and argument identification to permit learning in a simplified SRL system. Children do not

have the luxury of treating part-of-speech tagging and semantic role labeling as separable tasks. Instead, they must learn to understand sentences starting from scratch, learning the meanings of some words, and using those words and their patterns of arrangement into sentences to bootstrap their way into more mature knowledge.

We have created a first step toward modeling this incremental process. We combined unsupervised parsing with minimal supervision to begin to identify arguments and predicates. An SRL classifier used simple representations built from these identified arguments to extract useful abstract patterns for classifying semantic roles. Our results suggest that multiple simple representations of sentence structure could co-exist in the child's system for sentence comprehension; representations that will ultimately turn out to be powerful guides to role identification may be less powerful early in acquisition because of the noise introduced by the unsupervised parsing.

The next step is to 'close the loop', using higher level semantic feedback to improve the earlier argument identification and parsing stages. Perhaps with the help of semantic feedback the system can automatically improve predicate identification, which in turn allows it to correct the observed intransitive sentence error. This approach will move us closer to the goal of using initial simple structural patterns and natural observation of the world (semantic feedback) to bootstrap more and more sophisticated representations of linguistic structure.

Acknowledgments

This research is supported by NSF grant BCS-0620257 and NIH grant R01-HD054448.

References

M.J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

L. Bloom. 1970. *Language development: Form and function in emerging grammars*. MIT Press, Cambridge, MA.

L. Bloom. 1973. *One word at a time: The use of single-word utterances before syntax*. Mouton, The Hague.

M.R. Brent and J.M. Siskind. 2001. The role of exposure to isolated words in early vocabulary development. *Cognition*, 81:31–44.

E. Brill. 1997. Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora*. Kluwer Academic Press.

R. Brown. 1973. *A First Language*. Harvard University Press, Cambridge, MA.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared tasks: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. National Conference on Artificial Intelligence*.

E.V. Clark. 1978. Awwareness of language: Some evidence from what children say and do. In R. J. A. Sinclair and W. Levelt, editors, *The child's conception of language*. Springer Verlag, Berlin.

M. Connor, Y. Gertner, C. Fisher, and D. Roth. 2008. Baby srl: Modeling early language acquisition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages xx–yy, Aug.

M. Connor, Y. Gertner, C. Fisher, and D. Roth. 2009. Minimally supervised model of early language acquisition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, Jun.

M. Demetras, K. Post, and C. Snow. 1986. Feedback to first-language learners. *Journal of Child Language*, 13:275–292.

K. Demuth, J. Culbertson, and J. Alter. 2006. Word-minimality, epenthesis, and coda licensing in the acquisition of english. *Language & Speech*, 49:137–174.

C. Fisher. 1996. Structural limits on verb mapping: The role of analogy in children's interpretation of sentences. *Cognitive Psychology*, 31:41–81.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of EMNLP-2008*, pages 344–352.

D. Gentner. 2006. Why verbs are hard to learn. In K. Hirsh-Pasek and R. Golinkoff, editors, *Action meets word: How children learn verbs*, pages 544–564. Oxford University Press.

Y. Gertner and C. Fisher. 2006. Predicted errors in early verb learning. In *31st Annual Boston University Conference on Language Development*.

Y. Gertner, C. Fisher, and J. Eisengart. 2006. Learning words and rules: Abstract knowledge of word order in early sentence comprehension. *Psychological Science*, 17:684–691.

J. Gillette, H. Gleitman, L. R. Gleitman, and A. Lederer. 1999. Human simulations of vocabulary learning. *Cognition*, 73:135–176.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of 45th Annual Meeting of the Association of Computational Linguists*, pages 744–751.

R. Gomez and L. Gerken. 1999. Artificial grammar learning by 1-year-olds leads to specific and abstract knowledge. *Cognition*, 70:109–135.

A. Haghighi and D. Klein. 2006. Prototype-drive learning for sequence models. In *Proceedings of NAACL-2006*, pages 320–327.

Mark Johnson. 2007. Why doesnt em find good hmm pos-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

M.H. Kelly. 1992. Using sound to solve syntactic problems: The role of phonology in grammatical category assignments. *Psychological Review*, 99:349–364.

J. Lidz, H. Gleitman, and L. R. Gleitman. 2003. Understanding how input matters: verb learning and the footprint of universal grammar. *Cognition*, 87:151–178.

B. MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk. Third Edition*. Lawrence Elrbaum Associates, Mahwah, NJ.

M. P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

Marina Meilă. 2002. Comparing clusterings. Technical Report 418, University of Washington Statistics Department.

T. Mintz. 2003. Frequent frames as a cue for grammatical categories in child directed speech. *Cognition*, 90:91–117.

P. Monaghan, N. Chater, and M.H. Christiansen. 2005. The differential role of phonological and distributional cues in grammatical categorisation. *Cognition*, 96:143–182.

S. Pinker. 1984. *Language learnability and language development*. Harvard University Press, Cambridge, MA.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conferenceof the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

J.R. Saffran, R.N. Aslin, and E.L. Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274:1926–1928.

Rushen Shi, James L. Morgan, and Paul Allopenna. 1998. Phonological and acoustic bases for earliest grammatical category assignment: a cross-linguistic perspective. *Journal of Child Language*, 25(01):169–201.

Rushen Shi, Janet F. Werker, and James L. Morgan. 1999. Newborn infants' sensitivity to perceptual cues to lexical and grammatical words. *Cognition*, 72(2):B11 – B21.

L.B. Smith and C. Yu. 2008. Infants rapidly learn word-referent mappings via cross-situational statistics. *Cognition*, 106:1558–1568.

Kiristina Toutanova and Mark Johnson. 2007. A bayesian lda-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS*.

S. Yuan and C. Fisher. 2009. "really? she blicked the baby?": Two-year-olds learn combinatorial facts about verbs by listening. *Psychological Science*, 20:619–626.

S. Yuan, C. Fisher, Y. Gertner, and J. Snedeker. 2007. Participants are more than physical bodies: 21-month-olds assign relational meaning to novel transitive verbs. In *Biennial Meeting of the Society for Research in Child Development*, Boston, MA.

Modeling Norms of Turn-Taking in Multi-Party Conversation

Kornel Laskowski
Carnegie Mellon University
Pittsburgh PA, USA
kornel@cs.cmu.edu

Abstract

Substantial research effort has been invested in recent decades into the computational study and automatic processing of multi-party conversation. While most aspects of conversational speech have benefited from a wide availability of analytic, computationally tractable techniques, only qualitative assessments are available for characterizing multi-party turn-taking. The current paper attempts to address this deficiency by first proposing a framework for computing turn-taking model perplexity, and then by evaluating several multi-participant modeling approaches. Experiments show that direct multi-participant models do not generalize to held out data, and likely never will, for practical reasons. In contrast, the Extended-Degree-of-Overlap model represents a suitable candidate for future work in this area, and is shown to successfully predict the distribution of speech in time and across participants in previously unseen conversations.

1 Introduction

Substantial research effort has been invested in recent decades into the computational study and automatic processing of multi-party conversation. Whereas sociolinguists might argue that multi-party settings provide for the most natural form of conversation, and that dialogue and monologue are merely degenerate cases (Jaffe and Feldstein, 1970), computational approaches have found it most expedient to leverage past successes; these often involved at most one speaker. Consequently, even in multi-party settings, automatic systems generally continue to treat participants independently, fusing information across participants relatively late in processing.

This state of affairs has resulted in the near-exclusion from computational consideration and from semantic analysis of a phenomenon which occurs at the lowest level of speech exchange, namely the relative timing of the deployment of speech in arbitrary multi-party groups. This phenomenon, the implicit taking of turns at talk (Sacks et al., 1974), is important because unless participants adhere to its general rules, a conversation would simply not take place. It is therefore somewhat surprising that while most other aspects of speech enjoy a large base of computational methodologies for their study, there are few quantitative techniques for assessing the flow of turn-taking in general multi-party conversation.

The current work attempts to address this problem by proposing a simple framework, which, at least conceptually, borrows quite heavily from the standard language modeling paradigm. First, it defines the perplexity of a vector-valued Markov process whose multi-participant states are a concatenation of the binary states of individual speakers. Second, it presents some obvious evidence regarding the unsuitability of models defined directly over this space, under various assumptions of independence, for the inference of conversation-independent norms of turn-taking. Finally, it demonstrates that the extended-degree-of-overlap model of (Laskowski and Schultz, 2007), which models participants in an alternate space, achieves by far the best likelihood estimates for previously unseen conversations. This appears to be because the model can learn *across* conversations, regardless of the number of their participants. Experimental results show that it yields relative perplexity reductions of approximately 75% when compared to the ubiquitous single-participant model which ignores interlocutors, indicating that it can learn and generalize aspects of interaction which direct multi-participant models, and merely single-participant models, cannot.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 999–1008,
Uppsala, Sweden, 11-16 July 2010.

2 Data

Analysis and experiments are performed using the ICSI Meeting Corpus (Janin et al., 2003; Shriberg et al., 2004). The corpus consists of 75 meetings, held by various research groups at ICSI, which would have occurred even if they had not been recorded. This is important for studying naturally occurring interaction, since any form of intervention (including occurrence staging solely for the purpose of obtaining a record) may have an unknown but consistent impact on the emergence of turn-taking behaviors. Each meeting was attended by 3 to 9 participants, providing a wide variety of possible interaction types.

3 Conceptual Framework

3.1 Definitions

Turn-taking is a generally observed phenomenon in conversation (Sacks et al., 1974; Goodwin, 1981; Schegloff, 2007); one party talks while the others listen. Its description and analysis is an important problem, treated frequently as a subdomain of linguistic pragmatics (Levinson, 1983). In spite of this, linguists tend to disagree about what precisely constitutes a *turn* (Sacks et al., 1974; Edelsky, 1981; Goodwin, 1981; Traum and Heeman, 1997), or even a turn boundary. For example, a "yeah" produced by a listener to indicate attentiveness, referred to as a *backchannel* (Yngve, 1970), is often considered to not implement a turn (nor to delineate an ongoing turn of an interlocutor), as it bears no propositional content and does not "take the floor" from the current speaker.

To avoid being tied to any particular sociolinguistic theory, the current work equates "turn" with any contiguous interval of speech uttered by the same participant. Such intervals are commonly referred to as *talk spurts* (Norwine and Murphy, 1938). Because Norwine and Murphy's original definition is somewhat ambiguous and non-trivial to operationalize, this work relies on that proposed by (Shriberg et al., 2001), in which *spurts* are "defined as *speech regions uninterrupted by pauses longer than 500 ms*" (italics in the original). Here, a threshold of 300 ms is used instead, as recently proposed in NIST's Rich Transcription Meeting Recognition evaluations (NIST, 2002). The resulting definition of talk spurt, it is important to note, is in quite common use but frequently under different names. An oft-cited example is the *inter-pausal unit* of (Koiso et al., 1998)[1], where the threshold is 100 ms.

A consequence of this choice is that any *model of turn-taking behavior* inferred will effectively be a model of the distribution of speech, in time and across participants. If the parameters of such a model are maximum likelihood (ML) estimates, then that model will best account for what is most likely, or most "normal"; it will constitute a *norm*.

Finally, an important aspect of this work is that it analyzes turn-taking behavior as independent of the words spoken (and of the ways in which those words are spoken). As a result, strictly speaking, what is modeled is not the distribution of speech in time and across participants but of binary *speech activity* in time and across participants. Despite this seemingly dramatic simplification, it will be seen that important aspects of turn-taking are sufficiently rare to be problematic for modeling. Modeling them jointly alongside lexical information, in multi-party scenarios, is likely to remain intractable for the foreseeable future.

3.2 The Vocal Interaction Record Q

The notation used here, as in (Laskowski and Schultz, 2007), is a trivial extension of that proposed in (Rabiner, 1989) to vector-valued Markov processes.

At any instant t, each of K participants to a conversation is in a state drawn from $\Psi \equiv \{S_0, S_1\} \equiv \{\square, \blacksquare\}$, where $S_1 \equiv \blacksquare$ indicates speech (or, more precisely, "*intra*-talk-spurt instants") and $S_0 \equiv \square$ indicates non-speech (or "*inter*-talk-spurt instants"). The *joint* state of all participants at time t is described using the K-length column vector

$$\begin{aligned} \mathbf{q}_t \in \Psi^K &\equiv \Psi \times \Psi \times \ldots \times \Psi \\ &\equiv \left\{\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{2^K-1}\right\} . \end{aligned} \quad (1)$$

An entire conversation, from the point of view of this work, can be represented as the matrix

$$\begin{aligned} \mathbf{Q} &\equiv [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_T] \\ &\in \Psi^{K \times T} . \end{aligned} \quad (2)$$

$\mathbf{Q}$ is known as the (discrete) vocal interaction (Dabbs and Ruback, 1987) record. T is the total number of frames in the conversation, sampled at $T_s = 100$ ms intervals. This is approximately the duration of the shortest lexical productions in the ICSI Meeting Corpus.

[1] The inter-pausal unit differs from the *pause unit* of (Seligman et al., 1997) in that the latter is an intra-turn unit, requiring prior turn segmentation

3.3 Time-Independent First-Order Markov Modeling of $\mathbf{Q}$

Given this definition of $\mathbf{Q}$, a model $\mathbf{\Theta}$ is sought to account for it. Only time-independent models, whose parameters do not change over the course of the conversation, are considered in this work.

For simplicity, the state $\mathbf{q}_0 = \mathbf{S}_0 = [\square, \square, \ldots, \square]^*$, in which no participant is speaking (* indicates matrix transpose, to avoid confusion with conversation duration T) is first prepended to $\mathbf{Q}$. $P_0 = P(\mathbf{q}_0)$ therefore represents the unconditional probability of all participants being silent just prior to the start of any conversation[2]. Then

$$\begin{aligned} P(\mathbf{Q}) &= P_0 \cdot \prod_{t=1}^{T} P(\mathbf{q}_t \mid \mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_{t-1}) \\ &\doteq P_0 \cdot \prod_{t=1}^{T} P(\mathbf{q}_t \mid \mathbf{q}_{t-1}, \mathbf{\Theta}) \ , \end{aligned} \quad (3)$$

where in the second line the history is truncated to yield a standard first-order Markov form.

Each of the T factors in Equation 3 is independent of the instant t,

$$\begin{aligned} & P(\mathbf{q}_t \mid \mathbf{q}_{t-1}, \mathbf{\Theta}) \\ &\quad = P(\mathbf{q}_t = \mathbf{S}_j \mid \mathbf{q}_{t-1} = \mathbf{S}_i, \mathbf{\Theta}) \quad (4) \\ &\quad \equiv a_{ij} \ , \quad (5) \end{aligned}$$

as per the notation in (Rabiner, 1989). In particular, each factor is a function only of the state $\mathbf{S}_i$ in which the conversation was at time $t-1$ and the state $\mathbf{S}_j$ in which the conversation is at time t, and not of the instants $t-1$ or t. It may be expressed as the scalar a_{ij} which forms the ith row and jth column entry of the matrix $\{a_{ij}\} \equiv \mathbf{\Theta}$.

3.4 Perplexity

In language modeling practice, one finds the likelihood $P(\mathbf{w} \mid \mathbf{\Theta})$, of a word sequence $\mathbf{w}$ of length $\|\mathbf{w}\|$ under a model $\mathbf{\Theta}$, to be an inconvenient measure for comparison. Instead, the *negative log-likelihood* (NLL) and *perplexity* (PPL), defined as

$$\mathrm{NLL} = -\frac{1}{\|\mathbf{w}\|} \log_e P(\mathbf{w} \mid \mathbf{\Theta}) \quad (6)$$

$$\mathrm{PPL} = 10^{\mathrm{NLL}} \ , \quad (7)$$

[2] In reality, the instant $t = 0$ refers to the beginning of *the recording* of a conversation, rather than the beginning of the conversation itself; this detail is without consequence.

are often preferred (Jelinek, 1999). They are ubiquitously used to compare the complexity of different word sequences (or corpora) $\mathbf{w}$ and $\mathbf{w}'$ under the same model $\mathbf{\Theta}$, or the performance on a single word sequence (or corpus) $\mathbf{w}$ under competing models $\mathbf{\Theta}$ and $\mathbf{\Theta}'$.

Here, a similar metric is proposed, to be used for the same purposes, for the record $\mathbf{Q}$.

$$\mathrm{NLL} = -\frac{1}{KT} \log_2 P(\mathbf{Q} \mid \mathbf{\Theta}) \quad (8)$$

$$\begin{aligned} \mathrm{PPL} &= 2^{\mathrm{NLL}} \\ &= (P(\mathbf{Q} \mid \mathbf{\Theta}))^{-1/KT} \end{aligned} \quad (9)$$

are defined as measures of *turn-taking perplexity*. As can be seen in Equation 8, the negative log-likelihood is normalized by the number K of participants and the number T of frames in $\mathbf{Q}$; the latter renders the measure useful for making duration-independent comparisons. The normalization by K does not *per se* suggest that turn-taking in conversations with different K is necessarily similar; it merely provides similar bounds on the magnitudes of these metrics.

4 Direct Estimation of $\mathbf{\Theta}$

Direct application of bigram modeling techniques, defined over the states $\{\mathbf{S}\}$, is treated as a baseline.

4.1 The Case of $K = 2$ Participants

In contrast to multi-party conversation, dialogue has been extensively modeled in the ways described in this paper. Beginning with (Brady, 1969), Markov modeling techniques over the joint speech activity of two interlocutors have been explored by both the sociolinguist and the psycholinguist community (Jaffe and Feldstein, 1970; Dabbs and Ruback, 1987). The same models have also appeared in dialogue systems (Raux, 2008). Most recently, they have been augmented with duration models in a study of the Switchboard corpus (Grothendieck et al., 2009).

4.2 The Case of $K > 2$ Participants

In the general case beyond dialogue, such models have found less traction. This is partly due to the exponential growth in the number of states as K increases, and partly due to difficulties in interpretation. The only model for arbitrary K that the author is familiar with is the GroupTalk model (Dabbs and Ruback, 1987), which is unsuitable for the purposes here as it does not scale (with K,

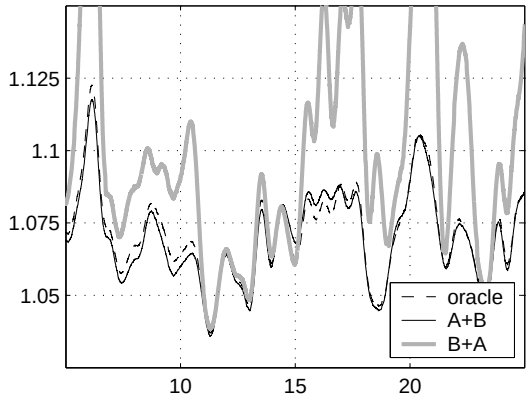


Figure 1: Perplexity (along y-axis) in time (along x-axis, in minutes) for meeting `Bmr024` under a conditionally dependent global oracle model, two "matched-half" models (A+B), and two "mismatched-half" models (B+A).

the number of participants) without losing track of speakers when two or more participants speak simultaneously (known as *overlap*).

4.2.1 Conditionally Dependent Participants

In a particular conversation with K participants, the state space of an ergodic process contains 2^K states, and the number of free parameters in a model $\mathbf{\Theta}$ which treats participant behavior as *conditionally dependent* (CD), henceforth $\mathbf{\Theta}^{CD}$, scales as $2^K \cdot \left(2^K - 1\right)$. It should be immediately obvious that many of the 2^K states are likely to not occur within a conversation of duration T, leading to misestimation of the desired probabilities.

To demonstrate this, three perplexity trajectories for a snippet of meeting `Bmr024` are shown in Figure 1, in the interval beginning 5 minutes into the meeting and ending 20 minutes later. (The meeting is actually just over 50 minutes long but only a snippet is shown to better appreciate small time-scale variation.) The depicted perplexities are not unweighted averages over the whole meeting of duration T as in Equation 8, but over a 60-second Hamming window centered on each t.

The first trajectory, the dashed black line, is obtained when the entire meeting is used to estimate $\mathbf{\Theta}^{CD}$, and is then scored by that same model (an "oracle" condition). Significant perplexity variation is observed throughout the depicted snippet.

The second trajectory, the continuous black line, is that obtained when the meeting is split into two equal-duration halves, one consisting of all instants prior to the midpoint and the other of all instants following it. These halves are hereafter referred to as A and B, respectively (the interval in Figure 1 falls entirely within the A half). Two separate models $\mathbf{\Theta}_A^{CD}$ and $\mathbf{\Theta}_B^{CD}$ are each trained on only one of the two halves, and then applied to those same halves. As can be seen at the scale employed, the matched A+B model, demonstrating the effect of training data ablation, deviates from the global oracle model only in the intervals $[7, 11]$ seconds and $[15, 18]$ seconds; otherwise it appears that more training data, from later in the conversation, does not affect model performance.

Finally, the third trajectory, the continuous gray line, is obtained when the two halves A and B of the meeting are scored using the mismatched models $\mathbf{\Theta}_B^{CD}$ and $\mathbf{\Theta}_A^{CD}$, respectively (this condition is henceforth referred to as the B+A condition). It can be seen that even when probabilities are estimated from the same participants, in exactly the same conversation, a direct conditionally dependent model exposed to over 25 minutes of a conversation cannot predict the turn-taking patterns observed later.

4.2.2 Conditionally Independent Participants

A potential reason for the gross misestimation of $\mathbf{\Theta}^{CD}$ under mismatched conditions is the size of the state space $\{\mathbf{S}\}$. The number of parameters in the model can be reduced by assuming that participants behave *independently* at instant t, but are conditioned on their *joint* behavior at $t-1$. The likelihood of $\mathbf{Q}$ under the resulting *conditionally independent model* $\mathbf{\Theta}^{CI}$ has the form

$$P\left(\mathbf{Q}\right) \doteq P_0 \cdot \prod_{t=1}^{T} \prod_{k=1}^{K} P\left(\mathbf{q}_t\left[k\right] \mid \mathbf{q}_{t-1}, \mathbf{\Theta}_k^{CI}\right) , \quad (10)$$

where each factor is time-independent,

$$P\left(\mathbf{q}_t\left[k\right] \mid \mathbf{q}_{t-1}, \mathbf{\Theta}_k^{CI}\right) = P\left(\mathbf{q}_t\left[k\right] = S_n \mid \mathbf{q}_{t-1} = \mathbf{S}_i, \mathbf{\Theta}_k^{CI}\right) \quad (11)$$

$$\equiv a_{k,in}^{CI} , \quad (12)$$

with $0 \leq i < 2^K$ and $0 \leq n < 2$. The complete model $\{\mathbf{\Theta}_k^{CI}\} \equiv \{\{a_{k,in}^{CI}\}\}$ consists of K matrices of size $2^K \times 2$ each. It therefore contains only $K \cdot 2^K$ free parameters, a significant reduction over the conditionally dependent model $\mathbf{\Theta}^{CD}$.

Panel (a) of Figure 2 shows the performance of this model on the same conversational snippet

as in Figure 1. The oracle, dashed black line of the latter is reproduced as a reference. The continuous black and gray lines show the smoothed perplexity for the matched (A+B) and the mismatched (B+A) conditions, respectively. In the matched condition, the CI model reproduces the oracle trajectory with relatively high fidelity, suggesting that participants' behavior may in fact be assumed to be conditionally independent in the sense discussed. Furthermore, the failures of the CI model under mismatched conditions are less severe in magnitude than those of the CD model.

Panel (b) of Figure 2 demonstrates the trivial fact that a conditionally independent model $\mathbf{\Theta}^{CI}_{any}$, tying the statistics of all K participants into a single model, is useless. This is of course because it cannot predict the next state of a generic participant for which the index k in $\mathbf{q}_{t-1}$ has been lost.

4.2.3 Mutually Independent Participants

A further reduction in the complexity of $\mathbf{\Theta}$ can be achieved by assuming that participants are *mutually independent* (MI), leading to the participant-specific $\mathbf{\Theta}^{MI}_k$ model:

$$P(\mathbf{Q}) \doteq P_0 \cdot \prod_{t=1}^{T} \prod_{k=1}^{K} P\left(\mathbf{q}_t[k] \mid \mathbf{q}_{t-1}[k], \mathbf{\Theta}^{MI}_k\right). \quad (13)$$

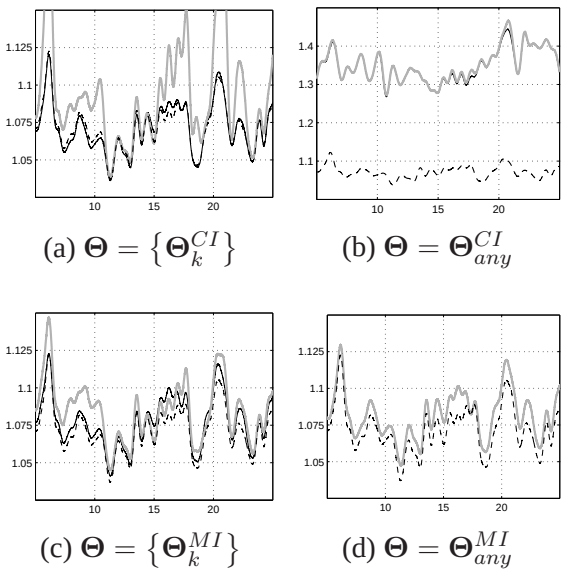
The factors are time-independent,

$$P\left(\mathbf{q}_t[k] \mid \mathbf{q}_{t-1}[k], \mathbf{\Theta}^{MI}_k\right) = P\left(\mathbf{q}_t[k] = S_n \mid \mathbf{q}_{t-1}[k] = S_m, \mathbf{\Theta}^{MI}_k\right) \quad (14)$$

$$\equiv a^{MI}_{k,mn}, \quad (15)$$

where $0 \leq m < 2$ and $0 \leq n < 2$. This model $\{\mathbf{\Theta}^{MI}_k\} \equiv \{\{a^{MI}_{k,mn}\}\}$ consists of K matrices of size 2×2 each, with only $K \cdot 2$ free parameters.

Panel (c) of Figure 2 shows that the MI model yields mismatched performance which is a much better approximation to its performance under matched conditions. However, its matched performance is worse than that of CD and CI models. When a single MI model $\mathbf{\Theta}^{MI}_{any}$ is trained instead for all participants, as shown in panel (d), both of these effects are exaggerated. In fact, the performance of $\mathbf{\Theta}^{MI}_{any}$ in matched and mismatched conditions is almost identical. The consistently higher perplexity is obtained, as mentioned, by smoothing over 60-second windows, and therefore underestimates poor performance at specific instants (which occur frequently).



(a) $\mathbf{\Theta} = \left\{\mathbf{\Theta}^{CI}_k\right\}$ (b) $\mathbf{\Theta} = \mathbf{\Theta}^{CI}_{any}$

(c) $\mathbf{\Theta} = \left\{\mathbf{\Theta}^{MI}_k\right\}$ (d) $\mathbf{\Theta} = \mathbf{\Theta}^{MI}_{any}$

Figure 2: Perplexity (along y-axis) in time (along x-axis, in minutes) for meeting `Bmr024` under a conditionally dependent global oracle model, and various matched (A+B) and mismatched (B+A) model pairs with relaxed dependence assumptions. Legend as in Figure 1.

5 Limitations and Desiderata

As the analyses in Section 4 reveal, direct estimation can be useful under oracle conditions, namely when all of a conversation has been observed and the task is to find intervals where multi-participant behavior deviates significantly from its *conversation-specific* norm. The assumption of conditional independence among participants was argued to lead to negligible degradation in the detectability of these intervals. However, the assumption of mutual independence consistently leads to higher surprise by the model.

5.1 Predicting the Future Within Conversations

In the more interesting setting in which only a part of a conversation has been seen and the task is to limit the perplexity of what is still to come, direct estimation exhibits relatively large failures under both conditionally dependent and conditionally independent participant assumptions. This appears to be due to the size of the state space, which scales as 2^K with the number K of participants. In the case of general K, more conversational data may be sought, from exactly the same group of participants, but that approach appears likely to be

insufficient, and, for practical reasons[3], impossible. One would instead like to be able to use other conversations, also exhibiting participant interaction, to limit the perplexity of speech occurrence in the conversation under study.

Unfortunately, there are two reasons why direct estimation cannot be tractably deployed across conversations. The first is that the direct models considered here, with the exception of $\mathbf{\Theta}^{MI}_{any}$, are K-*specific*. In particular, the number and the identity of conditioning states are both functions of K, for $\mathbf{\Theta}^{CD}$ and $\{\mathbf{\Theta}^{CI}_k\}$; the models may also consist of K distinct submodels, as for $\{\mathbf{\Theta}^{CI}_k\}$ and $\{\mathbf{\Theta}^{MI}_k\}$. No techniques for computing the turn-taking perplexity in conversations with K participants, using models trained on conversations with $K' \neq K$, are currently available.

The second reason is that these models, again with the exception of $\mathbf{\Theta}^{MI}_{any}$, are $\mathbf{R}$-*specific*, independently of K-specificity. By this it is meant that the models are sensitive to participant index permutation. Had a participant at index k in $\mathbf{Q}$ been assigned to another index $k'{\neq}k$, an alternate representation of the conversation, namely $\mathbf{Q}' = \mathbf{R}_{kk'} \cdot \mathbf{Q}$, would have been obtained. (Here, $\mathbf{R}_{kk'}$ is a matrix rotation operator obtained by exchanging columns k and k' of the $K \times K$ identity matrix $\mathbf{I}$.) Since index assignment is entirely arbitrary, useful direct models cannot be inferred from other conversations, even when their $K' = K$, unless K is small. The prospect of naively permuting every training conversation prior to parameter inference has complexity $K!$.

5.2 Comparing Perplexity Across Conversations

Until $\mathbf{R}$-specificity is comprehensively addressed, the only model from among those discussed so far, which exhibits no K-dependence, is $\mathbf{\Theta}^{MI}_{any}$, namely that which treats participants identically and independently. This model can be used to score the perplexity of *any* conversation, and facilitates the comparison of the distribution of speech activity *across* conversations.

Unfortunately, since the model captures only durational aspects of *one*-participant speech and non-speech intervals, it does not in any way encode a norm of turn-taking, an inherently interactive and hence *multi*-participant phenomenon. It therefore cannot be said to rank conversations according to their deviation from turn-taking norms.

[3]This pertains to the practicalities of re-inviting, instrumenting, recording and transcribing the same groups of participants, with necessarily more conversations for large groups than for small ones.

5.3 Theoretical Limitations

In addition to the concerns above, a fundamental limitation of the analyzed direct models, whether for conversation-specific or conversation-independent use, is that they are theoretically cumbersome if not vacuous. Given a solution to the problem of $\mathbf{R}$-specificity, the parameters $\{a^{CD}_{ij}\}$ may be robustly inferred, and the models may be applied to yield useful estimates of turn-taking perplexity. However, they cannot be said to directly validate or dispute the vast qualitative observations of sociolinguistics, and of conversation analysis in particular.

5.4 Prospects for Smoothing

To produce Figures 1 and 2, a small fraction of probability mass was reserved for unseen bigram transitions (as opposed to backing off to unigram probabilities). Furthermore, transitions into never-observed states were assigned uniform probabilities. This policy is simplistic, and there is significant scope for more detailed back-off and interpolation. However, such techniques infer values for under-estimated probabilities from *shorter truncations of the conditioning history*. As K-specificity and $\mathbf{R}$-specificity suggest, what appears to be needed here are back-off and interpolation *across states*. For example, in a conversation of $K = 5$ participants, estimates of the likelihood of the state $\mathbf{q}_t = [\square\blacksquare\blacksquare\blacksquare\square]^*$, which might have been unobserved in any training material, can be assumed to be related to those of $\mathbf{q}'_t = [\square\square\blacksquare\blacksquare\square]^*$ and $\mathbf{q}''_t = [\square\blacksquare\blacksquare\square\square]^*$, as well as those of $\mathbf{R}\mathbf{q}'_t$ and $\mathbf{R}\mathbf{q}''_t$, for arbitrary $\mathbf{R}$.

6 The Extended-Degree-of-Overlap Model

The limitations of direct models appear to be addressable by a form proposed by Laskowski and Schultz in (2006) and (2007). That form, the Extended-Degree-of-Overlap (EDO) model, was used to provide prior probabilities $P(\,\mathbf{Q}\,|\,\mathbf{\Theta}\,)$ of the speech states of multiple meeting participants simultaneously, for use in speech activity detection. The model was trained on *utterances* (rather than talk spurts) from a different corpus than that

used here, and the authors did not explore the turn-taking perplexities of their data sets.

Several of the equations in (Laskowski and Schultz, 2007) are reproduced here for comparison. The EDO model yields time-independent transition probabilities which assume conditional inter-participant dependence (cf. Equation 3),

$$P\left(\mathbf{q}_{t+1}=\mathbf{S}_j \,|\, \mathbf{q}_t=\mathbf{S}_i\right) \;=\; \alpha_{ij} \;\cdot \qquad (16)$$
$$P\left(\|\mathbf{q}_{t+1}\|=n_j,\, \|\mathbf{q}_{t+1}\cdot\mathbf{q}_t\|=o_{ij} \,|\, \|\mathbf{q}_t\|=n_i\right),$$

where $n_i \equiv \|\mathbf{S}_i\|$ and $n_j \equiv \|\mathbf{S}_j\|$, with $\|\mathbf{S}\|$ yielding the number of participants in ■ in the multi-participant state $\mathbf{S}$. In other words, n_i and n_j are the *numbers* of participants simultaneously speaking in states $\mathbf{S}_i$ and $\mathbf{S}_j$, respectively. The elements of the binary product $\mathbf{S}=\mathbf{S}_1\cdot\mathbf{S}_2$ are given by

$$\mathbf{S}\left[k\right] \;\equiv\; \begin{cases} \blacksquare, & \text{if } \mathbf{S}_1\left[k\right]=\mathbf{S}_2\left[k\right]=\blacksquare \\ \square, & \text{otherwise}, \end{cases} \qquad (17)$$

and o_{ij} is therefore the number of same participants speaking in $\mathbf{S}_i$ *and* $\mathbf{S}_j$. The discussion of the role of α_{ij} in Equation 16 is deferred to the end of this section.

The EDO model mitigates $\mathbf{R}$-specificity because it models each bigram $(\mathbf{q}_{t-1},\mathbf{q}_t)=(\mathbf{S}_i,\mathbf{S}_j)$ as the modified bigram $(n_i,[o_{ij},n_j])$, involving three scalars each of which is a *sum* — a commutative (and therefore rotation-invariant) operation. Because it sums across only those participants which are in the ■ state, completely ignoring their □-state interlocutors, it can also mitigate K-specificity if one additionally redefines

$$n_i \;=\; \min\left(\|\mathbf{S}_i\|,\, K_{max}\right) \qquad (18)$$
$$n_j \;=\; \min\left(\|\mathbf{S}_j\|,\, K_{max}\right) \qquad (19)$$
$$o_{ij} \;=\; \min\left(\|\mathbf{S}_i\cdot\mathbf{S}_j\|,\, n_i,\, n_j\right), \qquad (20)$$

as in (Laskowski and Schultz, 2007). K_{max} represents the maximum model-licensed degree of overlap, or the maximum number of participants allowed to be simultaneously speaking. The EDO model therefore represents a viable conversation-independent, K-independent, and $\mathbf{R}$-independent model of turn-taking for the purposes in the current work[4]. The factor α_{ij} in Equation 16 provides a deterministic mapping from the conversation-independent space $(n_i,[o_{ij},n_j])$ to the conversation-specific space $\{a_{ij}\}$. The mapping is deterministic because the model assumes that all participants are identical. This places the EDO model at a disadvantage with respect to the CD and CI models, as well as to $\{\mathbf{\Theta}_k^{MI}\}$, which allow each participant to be modeled differently.

7 Experiments

This section describes the performance of the discussed models on the entire ICSI Meeting Corpus.

7.1 Conversation-Specific Modeling

First to be explored is the prediction of yet-unobserved behavior in conversation-specific settings. For each meeting, models are trained on portions of that meeting only, and then used to score other portions of the same meeting. This is repeated over all meetings, and comprises the mismatched condition of Section 4; for contrast, the matched condition is also evaluated.

Each meeting is divided into two halves, in two different ways. The first way is the A/B split of Section 4, representing the first and second halves of each meeting; as has been shown, turn-taking patterns may vary substantially from A to B. The second split (C/D) places every even-numbered frame in one set and every odd-numbered frame in the other. This yields a much easier setting, of two halves which are on average maximally similar but still temporally disjoint.

The perplexities (of Equation 9) in these experiments are shown in the second, fourth, sixth and eighth columns of Table 1, under "all". In the matched A+B and C+D conditions, the conditionally dependent model $\mathbf{\Theta}^{CD}$ provides topline ML performance. Perplexities decrease as model complexities fall for direct models, as expected. However, in the more interesting mismatched B+A condition, the EDO model performs the best. This shows that its ability to generalize to unseen data is higher than that of direct models. However, in the easier mismatched D+C condition, it is outperformed by the CI model due to behavior differences among participants, which the EDO model

[4]There exists some empirical evidence to suggest that conversations of K participants should not be used to train models for predicting turn-taking behavior in conversations of K' participants, for $K'\neq K$, because turn-taking is inherently K-dependent. For example, (Fay et al., 2000) found that qualitative differences in turn-taking patterns between small groups and large groups, represented in their study by $K=5$ and $K=10$, and noted that there is a smooth transition between the two extremes; this provides some scope for interpolating small- and large- group models, and the EDO framework makes this possible.

Model	Hard split A/B (first/second halves)				Easy split C/D (odd/even frames)			
	A+B		B+A		C+D		D+C	
	"all"	"sub"	"all"	"sub"	"all"	"sub"	"all"	"sub"
$\mathbf{\Theta}^{CD}$	**1.0905**	**1.6444**	1.1225	1.8395	**1.0915**	**1.6555**	1.0991	1.7403
$\{\mathbf{\Theta}_k^{CI}\}$	1.0915	1.6576	1.1156	1.7809	1.0925	1.6695	**1.0956**	**1.7028**
$\{\mathbf{\Theta}_k^{MI}\}$	1.0978	1.7236	1.1086	1.7950	1.0991	1.7381	1.0992	1.7398
$\mathbf{\Theta}^{MI}$	1.1046	1.8047	1.1047	1.8059	1.1046	1.8050	1.1046	1.8052
$\mathbf{\Theta}^{EDO}$	1.0977	1.7257	**1.0985**	**1.7323**	1.0977	1.7268	1.0982	1.7313

Table 1: Perplexities for conversation-specific turn-taking models on the entire ICSI Meeting Corpus. Both "all" frames and the subset ("sub") for which $\mathbf{q}_{t-1} \neq \mathbf{q}_t$ are shown, for matched (A+B and C+D) and mismatched (B+A and D+C) conditions on splits A/B and C/D.

does not capture.

The numbers under the "all" columns in Table 1 were computed using all of each meeting's frames. For contrast, in the "sub" columns, perplexities are computed over only those frames for which $\mathbf{q}_{t-1} \neq \mathbf{q}_t$. This is a useful subset because, for the majority of time in conversations, one person simply continues to talk while all others remain silent[5]. Excluding $\mathbf{q}_{t-1} = \mathbf{q}_t$ bigrams (leading to 0.32M frames from 2.39M frames in "all") offers a glimpse of expected performance differences were duration modeling to be included in the models. Perplexities are much higher in these intervals, but the same general trend as for "all" is observed.

7.2 Conversation-Independent Modeling

The training of conversation-independent models, given a corpus of K-heterogeneous meetings, is achieved by iterating over all meetings and testing each using models trained on all of the other meetings. As discussed in the preceding section, $\mathbf{\Theta}_{any}^{MI}$ is the only one among the direct models which can be used for this purpose. It also models exclusively single-participant behavior, ignoring the interactive setting provided by other participants. As shown in Table 2, when all time is scored the EDO model with $K_{max} = 4$ is the best model (in Section 7.1, $K_{max} = K$ since the model was trained on the same meeting to which it was applied). Its perplexity gap to the oracle model is only a quarter of the gap exhibited by $\mathbf{\Theta}_{any}^{MI}$.

The relative performance of EDO models is even better when only those instants t are considered for which $\mathbf{q}_{t-1} \neq \mathbf{q}_t$. There, the perplexity gap to the oracle model is smaller than that of $\mathbf{\Theta}^{EDO}$ by 78%.

[5]Retaining only $\mathbf{q}_{t-1} \neq \mathbf{q}_t$ also retains instants of transition into and out of intervals of silence.

Model	PPL		ΔPPL (%)	
	"all"	"sub"	"all"	"sub"
$\mathbf{\Theta}^{CD}$	1.0921	1.6616	—	—
$\mathbf{\Theta}^{MI}$	1.1051	1.8170	14.1	23.5
$\mathbf{\Theta}^{EDO}$ (6)	1.0992	1.7405	7.7	11.9
$\mathbf{\Theta}^{EDO}$ (5)	1.0968	1.7127	5.1	7.7
$\mathbf{\Theta}^{EDO}$ (4)	**1.0953**	**1.6947**	3.5	5.0
$\mathbf{\Theta}^{EDO}$ (3)	1.1082	1.8502	17.5	28.5

Table 2: Perplexities for conversation-independent turn-taking models on the entire ICSI Meeting Corpus; the oracle $\mathbf{\Theta}^{CD}$ topline is included in the first row. Both "all" frames and the subset ("sub") for which $\mathbf{q}_{t-1} \neq \mathbf{q}_t$ are shown; relative increases over the topline (less unity, representing no perplexity) are shown in columns 4 and 5. The value of K_{max} (cf. Equations 18, 19, and 20) is shown in parentheses in the first column.

8 Discussion

The model perplexities as reported above may be somewhat different if the "talk spurt" were replaced by a more sociolinguistically motivated definition of "turn", but the ranking of models and their relative performance differences are likely to remain quite similar. On the one hand, many inter-talk-spurt gaps might find themselves to be within-turn, leading to more ■ entries in the record $\mathbf{Q}$ than observed in the current work. This would increase the apparent frequency and duration of intervals of overlap. On the other hand, alternative definitions of turn may exclude some speech activity, such as that implementing backchannels. Since backchannels are often produced in overlap

with the foreground speaker, their removal may eliminate some overlap from $\mathbf{Q}$. (However, as noted in (Shriberg et al., 2001), overlap rates in multi-party conversation remain high even after the exclusion of backchannels.) Both inter-talk-spurt gap inclusion and backchannel exclusion are likely to yield systematic differences, and therefore to be exploitable by the investigated models in similar ways.

The results presented may also be perturbed by modifying the way in which a (manually produced) talk spurt segmentation, with high-precision boundary time-stamps, is discretized to yield $\mathbf{Q}$. Two parameters have controlled the discretization in this work: (1) the frame step $T_s = 100$ ms; and (2) the proportion ρ of T_s for which a participant must be speaking within a frame in order for that frame to be considered ■ rather than □. $\rho = 0.5$ was chosen since this posits approximately as much more speech (than in the high-precision segmentation) as it eliminates. Higher values of ρ would lead to more ■, leading to more overlap than observed in this work. Meanwhile, at constant ρ, choosing a T_s value larger than 100 ms would occasionally miss the shortest talk spurts, but it would allow the models, which are all 1st-order Markovian, to learn temporally more distant dependencies. The trade-offs between these choices are currently under investigation.

From an operational, modeling perspective, it is important to recognize that the choices of the definition for "turn", and of the way in which segmentations are discretized, are essentially arbitrary. The investigated modeling alternatives, and the EDO model in particular, require only that the multi-participant vocal interaction record $\mathbf{Q}$ be *binary-valued*. This general applicability has been demonstrated in past work, in which the EDO model was trained on *utterances* for use in speech activity detection (Laskowski and Schultz, 2007), as well as in (Laskowski and Burger, 2007) where it was trained separately on talk spurts and *laugh bouts*, in the same data, to highlight the differences between speech and laughter deployment.

Finally, it should be remembered that the EDO model is both time-independent and participant-independent. This makes it suitable for comparison of conversational genres, in much the same way as are general language models of words. Accordingly, as for language models, density estimation in future turn-taking models may be improved by considering variability across participants and in time. Participant dependence is likely to be related to speakers' social characteristics and conversational roles, while time dependence may reflect opening and closing functions, topic boundaries, and periodic turn exchange failures. In the meantime, event types such as the latter may be detectable as EDO perplexity departures, potentially recommending the model's use for localizing conversational "hot spots" (Wrede and Shriberg, 2003). The EDO model, and turn-taking models in general, may also find use in diagnosing turn-taking naturalness in spoken dialogue systems.

9 Conclusions

This paper has presented a framework for quantifying the turn-taking perplexity in multi-party conversations. To begin with, it explored the consequences of modeling participants jointly by concatenating their binary speech/non-speech states into a single multi-participant vector-valued state. Analysis revealed that such models are particularly poor at generalization, even to subsequent portions of the same conversation. This is due to the size of their state space, which is factorial in the number of participants. Furthermore, because such models are both specific to the number of participants and to the order in which participant states are concatenated together, it is generally intractable to train them on material from other conversations. The only such model which may be trained on other conversations is that which completely ignores interlocutor interaction.

In contrast, the Extended-Degree-of-Overlap (EDO) construction of (Laskowski and Schultz, 2007) may be trained on other conversations, regardless of their number of participants, and usefully applied to approximate the turn-taking perplexity of an oracle model. This is achieved because it models entry into and egress out of specific degrees of overlap, and completely ignores the number of participants actually present or their modeled arrangement. In this sense, the EDO model can be said to implement the qualitative findings of conversation analysis. In predicting the distribution of speech in time and across participants, it reduces the unseen data perplexity of a model which ignores interaction by 75% relative to an oracle model.

References

Paul T. Brady. 1969. A model for generating on-off patterns in two-way conversation. *Bell Systems Technical Journal*, 48(9):2445–2472.

James M. Dabbs and R. Barry Ruback. 1987. Dimensions of group process: Amount and structure of vocal interaction. *Advances in Experimental Social Psychology*, 20:123–169.

Carole Edelsky. 1981. Who's got the floor? *Langauge in Society*, 10:383–421.

Nicolas Fay, Simon Garrod, and Jean Carletta. 2000. Group discussion as interactive dialogue or as serial monologue: The influence of group size. *Psychological Science*, 11(6):487–492.

Charles Goodwin. 1981. *Conversational Organization: Interaction Between Speakers and Hearers*. Academic Press, New York NY, USA.

John Grothendieck, Allen Gorin, and Nash Borges. 2009. Social correlates of turn-taking behavior. *Proc. ICASSP*, Taipei, Taiwan, pp. 4745–4748.

Joseph Jaffe and Stanley Feldstein. 1970. *Rhythms of Dialogue*. Academic Press, New York NY, USA.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The ICSI Meeting Corpus. *Proc. ICASSP*, Hong Kong, China, pp. 364–367.

Frederick Jelinek. 1999. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge MA, USA.

Hanae Koiso, Yasui Horiuchi, Syun Tutiya, Akira Ichikawa, and Yasuharu Den. 1998. An analysis of turn-taking and backchannels based on prosodic and syntactic features in Japanese Map Task dialogs. *Language and Speech*, 41(3-4):295–321.

Kornel Laskowski and Tanja Schultz. 2006. Unsupervised learning of overlapped speech model parameters for multichannel speech activity detection in meetings. *Proc. ICASSP*, Toulouse, France, pp. 993–996.

Kornel Laskowski and Susanne Burger. 2007. Analysis of the occurrence of laughter in meetings. *Proc. INTERSPEECH*, Antwerpen, Belgium, pp. 1258–1261.

Kornel Laskowski and Tanja Schultz. 2007. Modeling vocal interaction for segmentation in meeting recognition. *Machine Learning for Multimodal Interaction*, A. Popescu-Belis, S. Renals, and H. Bourlard, eds., Lecture Notes in Computer Science, 4892:259–270, Springer Berlin/Heidelberg, Germany.

Stephen C. Levinson. 1983. *Pragmatics*. Cambridge University Press.

National Institute of Standards and Technology. 2002. Rich Transcription Evaluation Project, `www.itl.nist.gov/iad/mig/tests/rt/` (last accessed 15 February 2010 1217hrs GMT).

A. C. Norwine and O. J. Murphy. 1938. Characteristic time intervals in telephonic conversation. *Bell System Technical Journal*, **17**:281-291.

Lawrence Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286.

Antoine Raux. 2008. Flexible turn-taking for spoken dialogue systems. PhD Thesis, Carnegie Mellon University.

Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. 1974. A simplest semantics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

Emanuel A. Schegloff. 2007. *Sequence Organization in Interaction*. Cambridge University Press, Cambridge, UK.

Mark Seligman, Junko Hosaka, and Harald Singer. 1997. "Pause units" and analysis of spontaneous Japanese dialogues: Preliminary studies. *Dialogue Processing in Spoken Language Systems* E. Maier, M. Mast, and S. LuperFoy, eds., Lecture Notes in Computer Science, 1236:100–112. Springer Berlin/Heidelberg, Germany.

Elizabeth Shriberg, Andreas Stolcke, and Don Baron. 2001. Observations on overlap: Findings and implications for automatic processing of multi-party conversation. *Proc. EUROSPEECH*, Genève, Switzerland, pp. 1359–1362.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The ICSI Meeting Recorder Dialog Act (MRDA) Corpus. *Proc. SIGDIAL*, Boston MA, USA, pp. 97–100.

David Traum and Peeter Heeman. 1997. Utterance units in spoken dialogue. *Dialogue Processing in Spoken Language Systems* E. Maier, M. Mast, and S. LuperFoy, eds., Lecture Notes in Computer Science, 1236:125–140. Springer Berlin/Heidelberg, Germany.

Britta Wrede and Elizabeth Shriberg. 2003. Spotting "hot spots" in meetings: Human judgments and prosodic cues. *Proc. EUROSPEECH*, Aalborg, Denmark, pp. 2805–2808.

Victor H. Yngve. 1970. On getting a word in edgewise. *Papers from the Sixth Regional Meeting Chicago Linguistic Society*, pp. 567–578. Chicago Linguistic Society, Chicago IL, USA.

Optimising Information Presentation for Spoken Dialogue Systems

Verena Rieser
University of Edinburgh
Edinburgh, United Kingdom
verena.rieser@ed.ac.uk

Oliver Lemon
Heriot-Watt University
Edinburgh, United Kingdom
o.lemon@hw.ac.uk

Xingkun Liu
Heriot-Watt University
Edinburgh, United Kingdom
x.liu@hw.ac.uk

Abstract

We present a novel approach to Information Presentation (IP) in Spoken Dialogue Systems (SDS) using a data-driven statistical optimisation framework for content planning and attribute selection. First we collect data in a Wizard-of-Oz (WoZ) experiment and use it to build a supervised model of human behaviour. This forms a baseline for measuring the performance of optimised policies, developed from this data using Reinforcement Learning (RL) methods. We show that the optimised policies significantly outperform the baselines in a variety of generation scenarios: while the supervised model is able to attain up to 87.6% of the possible reward on this task, the RL policies are significantly better in 5 out of 6 scenarios, gaining up to 91.5% of the total possible reward. The RL policies perform especially well in more complex scenarios. We are also the first to show that adding predictive "lower level" features (e.g. from the NLG realiser) is important for optimising IP strategies according to user preferences. This provides new insights into the nature of the IP problem for SDS.

1 Introduction

Work on evaluating SDS suggests that the Information Presentation (IP) phase is the primary contributor to dialogue duration (Walker et al., 2001), and as such, is a central aspect of SDS design. During this phase the system returns a set of items ("hits") from a database, which match the user's current search constraints. An inherent problem in this task is the trade-off between presenting "enough" information to the user (for example helping them to feel confident that they have a good overview of the search results) versus keeping the utterances short and understandable.

In the following we show that IP for SDS can be treated as a data-driven joint optimisation problem, and that this outperforms a supervised model of human 'wizard' behaviour on a particular IP task (presenting sets of search results to a user).

A similar approach has been applied to the problem of Referring Expression Generation in dialogue (Janarthanam and Lemon, 2010).

1.1 Previous work on Information Presentation in SDS

Broadly speaking, IP for SDS can be divided into two main steps: 1) IP strategy selection and 2) Content or Attribute Selection. Prior work has presented a variety of **IP strategies** for structuring information (see examples in Table 1). For example, the SUMMARY strategy is used to guide the user's "focus of attention". It draws the user's attention to relevant attributes by grouping the current results from the database into clusters, e.g. (Polifroni and Walker, 2008; Demberg and Moore, 2006). Other studies investigate a COMPARE strategy, e.g. (Walker et al., 2007; Nakatsu, 2008), while most work in SDS uses a RECOMMEND strategy, e.g. (Young et al., 2007). In a previous proof-of-concept study (Rieser and Lemon, 2009) we show that each of these strategies has its own strengths and drawbacks, dependent on the particular context in which information needs to be presented to a user. Here, we will also explore possible combinations of the strategies, for example SUMMARY followed by RECOMMEND, e.g. (Whittaker et al., 2002), see Figure 1.

Prior work on **Content or Attribute Selection** has used a "Summarize and Refine" approach (Polifroni and Walker, 2008; Polifroni and Walker, 2006; Chung, 2004). This method employs utility-based attribute selection with respect to how each attribute (e.g. price or food type in restaurant

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1009–1018,
Uppsala, Sweden, 11-16 July 2010.

search) of a set of items helps to narrow down the user's goal to a single item. Related work explores a user modelling approach, where attributes are ranked according to user preferences (Demberg and Moore, 2006; Winterboer et al., 2007). Our data collection (see Section 3) and training environment incorporate these approaches.

The work in this paper is the first to apply a data-driven method to this whole decision space (i.e. combinations of Information Presentation strategies as well as attribute selection), and to show the utility of both lower-level features (e.g. from the NLG realiser) and higher-level features (e.g. from Dialogue Management) for this problem. Previous work has only focused on individual aspects of the problem (e.g. how many attributes to generate, or when to use a SUMMARY), using a pipeline model for SDS with DM features as input, and where NLG has no knowledge of lower level features (e.g. behaviour of the realiser). In Section 4.3 we show that lower level features significantly influence users' ratings of IP strategies. In the following we use a Reinforcement Learning (RL) as a statistical planning framework (Sutton and Barto, 1998) to explore the contextual features for making these decisions, and propose a new joint optimisation method for IP strategies combining content structuring and attribute selection.

2 NLG as planning under uncertainty

We follow the overall framework of NLG as planning under uncertainty (Lemon, 2008; Rieser and Lemon, 2009; Lemon, 2010), where each NLG action is a sequential decision point, based on the current dialogue context and the expected long-term utility or "reward" of the action. Other recent approaches describe this task as planning, e.g. (Koller and Petrick, 2008), or as contextual decision making according to a cost function (van Deemter, 2009), but not as a statistical planning problem, where uncertainty in the stochastic environment is explicitly modelled. Below, we apply this framework to Information Presentation strategies in SDS using Reinforcement Learning, where the example task is to present a set of search results (e.g. restaurants) to users. In particular, we consider 7 possible policies for structuring the content (see Figure 1): Recommending one single item, comparing two items, summarising all of them, or ordered combinations of those actions, e.g. first summarise all the retrieved items and then recommend one of them. The IP module has to decide which action to take next, how many attributes to mention, and when to stop generating.

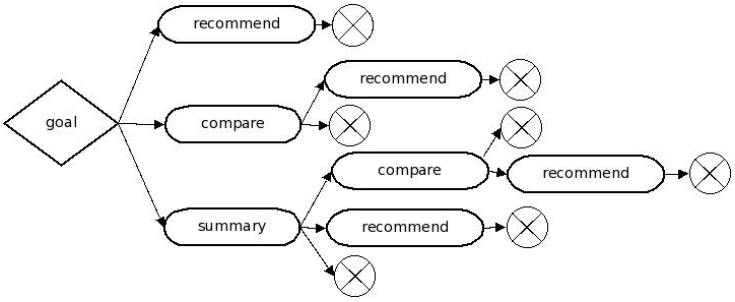


Figure 1: Possible Information Presentation structures (X=stop generation)

3 Wizard-of-Oz data collection

In an initial Wizard-of-Oz (WoZ) study, we asked humans (our "wizards") to produce good IP actions in different dialogue contexts, when interacting in spoken dialogues with other humans (the "users"), who believed that they were talking to an automated SDS. The wizards were experienced researchers in SDS and were familiar with the search domain (restaurants in Edinburgh). They were instructed to select IP structures and attributes for NLG so as to most efficiently allow users to find a restaurant matching their search constraints. They also received prior training on this task.

The task for the wizards was to decide which IP structure to use next (see Section 3.2 for a list of IP strategies to choose from), which attributes to mention (e.g. cuisine, price range, location, food quality, and/or service quality), and whether to stop generating, given varying numbers of database matches, varying prompt realisations, and varying user behaviour. Wizard utterances were synthesised using a state-of-the-art text-to-speech engine. The user speech input was delivered to the wizard using Voice Over IP. Figure 2 shows the web-based interface for the wizard.

3.1 Experimental Setup and Data collection

We collected 213 dialogues with 18 subjects and 2 wizards (Liu et al., 2009). Each user performed a total of 12 tasks, where no task set was seen twice by any one wizard. The majority of users were from a range of backgrounds in a higher education institute, in the age range 20-30, native speakers of English, and none had prior experience of

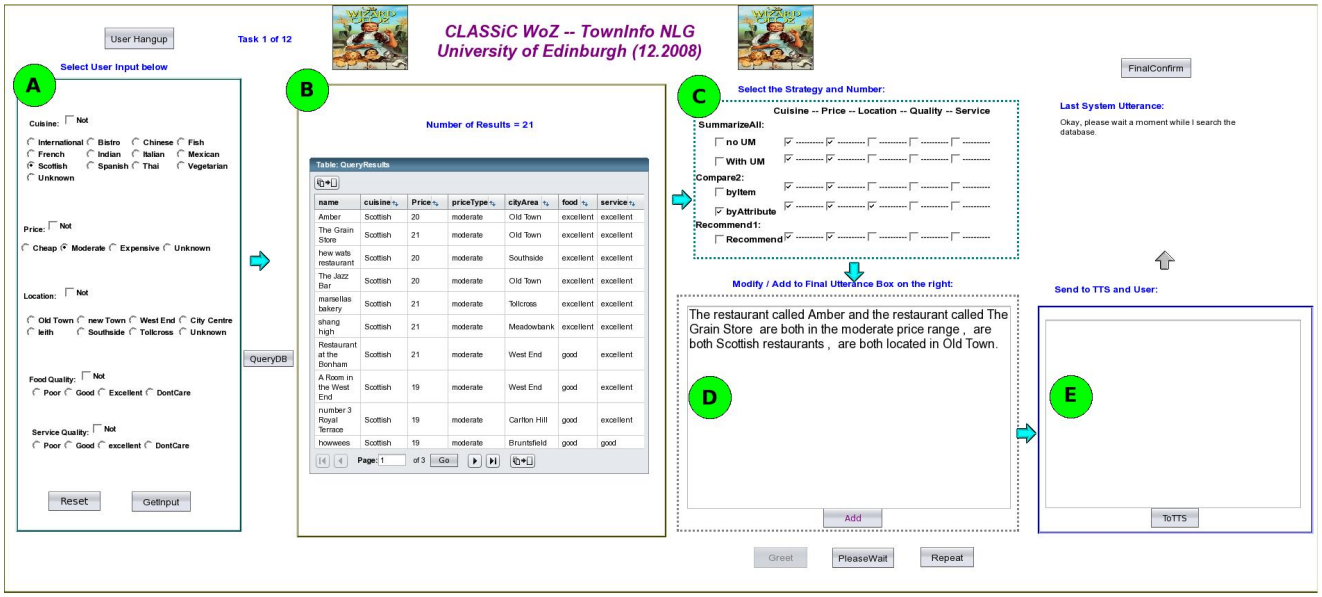


Figure 2: Wizard interface. [A:] The wizard selects attribute values as specified by the user's query. [B:] The retrieved database items are presented in an ordered list. We use a User Modelling approach for ranking the restaurants, see e.g. (Polifroni and Walker, 2008). [C:] The wizard then chooses which strategy and which attributes to generate next, by clicking radio buttons. The attribute/s specified in the last user query are pre-selected by default. The strategies can only be combined in the orders as specified in Figure 1. [D:] An utterance is automatically generated by the NLG realiser every time the wizard selects a strategy, and is displayed in an intermediate text panel. [E:] The wizard can decide to add the generated utterance to the final output panel or to start over again. The text in the final panel is sent to the user via TTS, once the wizard decides to stop generating.

Strategy	Example utterance
SUMMARY no UM	I found 26 restaurants, which have Indian cuisine. 11 of the restaurants are in the expensive price range. Furthermore, 10 of the restaurants are in the cheap price range and 5 of the restaurants are in the moderate price range.
SUMMARY UM	26 restaurants meet your query. There are 10 restaurants which serve Indian food and are in the cheap price range. There are also 16 others which are more expensive.
COMPARE by Item	The restaurant called Kebab Mahal is an Indian restaurant. It is in the cheap price range. And the restaurant called Saffrani, which is also an Indian restaurant, is in the moderate price range.
COMPARE by Attribute	The restaurant called Kebab Mahal and the restaurant called Saffrani are both Indian restaurants. However, Kebab Mahal is in the cheap price range while Saffrani is moderately priced.
RECOMMEND	The restaurant called Kebab Mahal has the best overall quality amongst the matching restaurants. It is an Indian restaurant, and it is in the cheap price range.

Table 1: Example realisations, generated when the user provided `cuisine=Indian`, and where the wizard has also selected the additional attribute `price` for presentation to the user.

Spoken Dialogue Systems. After each task the user answered a questionnaire on a 6 point Likert scale, regarding the perceived generation quality in that task. The wizards' IP strategies were highly ranked by the users on average (4.7), and users were able to select a restaurant in 98.6% of the cases. No significant difference between the wizards was observed.

The data contains 2236 utterances in total: 1465 wizard utterances and 771 user utterances. We automatically extracted 81 features (e.g #sentences, #DBhits, #turns, #ellipsis)[1] from the XML logfiles after each dialogue. Please see (Rieser et al., 2009) for more details.

[1]The full corpus and list of features is available at `https://www.classic-project.org/corpora/`

3.2 NLG Realiser

In the Wizard-of-Oz environment we implemented a NLG realiser for the chosen IP structures and attribute choices, in order to realise the wizards' choices in real time. This generator is based on data from the stochastic sentence planner SPaRKy (Stent et al., 2004). We replicated the variation observed in SPaRKy by analysing high-ranking example outputs (given the highest possible score by the SPaRKy judges) and implemented the variance using dynamic sentence generation. The realisations vary in sentence aggregation, aggregation operators (e.g. 'and', period, or ellipsis), contrasts

(e.g. 'however', 'on the other hand') and referring expressions (e.g. 'it', 'this restaurant') used. The length of an utterance also depends on the number of attributes chosen, i.e. the more attributes the longer the utterance. All of these variations were logged.

In particular, we realised the following IP strategies (see examples in Table 1):

- SUMMARY of all matching restaurants with or without a User Model (UM), following (Polifroni and Walker, 2008). The approach using a UM assumes that the user has certain preferences (e.g. cheap) and only tells him about the relevant items, whereas the approach with no UM lists all the matching items.
- COMPARE the top 2 restaurants by Item (i.e. listing all the attributes for the first item and then for the other) or by Attribute (i.e. directly comparing the different attribute values).
- RECOMMEND the top-ranking restaurant (according to UM).

Note that there was no discernible pattern in the data about the wizards' decisions between the UM/no UM and the byItem/byAttribute versions of the strategies. In this study we will therefore concentrate on the higher level decisions (SUMMARY VS. COMPARE VS. RECOMMEND) and model these different realisations as noise in the realiser.

3.3 Supervised Baseline strategy

We analysed the WoZ data to explore the best-rated strategies (the top scoring 50%, $n = 205$) that were employed by humans for this task. Here we used a variety of Supervised Learning methods to create a model of the highly rated wizard behaviour. Please see (Rieser et al., 2009) for further details. The best performing method was Rule Induction (JRip). [2] The model achieved an accuracy of 43.19% which is significantly ($p < .001$) better than the majority baseline of always choosing SUMMARY (34.65%). [3] The resulting rule set is shown in Figure 3.

```
IF (dbHits <= 9)& (prevNLG = summary):
   THEN nlgStrategy=compare;
IF (dbHits = 1):
   THEN nlgStrategy= Recommend;
IF(prevNLG=summaryRecommend)&(dbHits>=10):
   THEN nlgStrategy= Recommend;
ELSE nlgStrategy=summary;
```

Figure 3: Rules learned by JRip for the wizard model ('dbHits'= number of database matches, 'prevNLG'= previous NLG action)

The features selected by this model were only "high-level" features, i.e. the input (previous action, number of database hits) that an IP module receives as input from a Dialogue Manager (DM). We further analysed the importance of different features using feature ranking and selection methods (Rieser et al., 2009), finding that the human wizards in this specific setup did not pay significant attention to any lower level features, e.g. from surface realisation, although the generated output was displayed to them (see Figure 2).

Nevertheless, note that the supervised model achieves up to 87.6% of the possible reward on this task, as we show in Section 5.2, and so can be considered a serious baseline against which to measure performance. Below, we will show that Reinforcement Learning (RL) produces a significant improvement over the strategies present in the original data, especially in cases where RL has access to "lower level" features of the context.

4 The Simulation / Learning Environment

Here we "bootstrap" a simulated training environment from the WoZ data, following (Rieser and Lemon, 2008).

4.1 User Simulations

User Simulations are commonly used to train strategies for Dialogue Management, see for example (Young et al., 2007). A user simulation for NLG is very similar, in that it is a predictive model of the most likely next user act. [4] However, this NLG predicted user act does not actually change the overall dialogue state (e.g. by filling slots) but it only changes the generator state. In other words,

[2] The WEKA implementation of (Cohen, 1995)'s RIPPER.

[3] Note that the low accuracy is due to data sparsity and diverse behaviour of the wizards. However, in (Rieser et al., 2009) we show that this model is significantly different from the policy learned using the worse scoring 50%.

[4] Similar to the internal user models applied in recent work on POMDP (Partially Observable Markov Decision Process) dialogue managers (Young et al., 2007; Henderson and Lemon, 2008; Gasic et al., 2008) for estimation of user act probabilities.

the NLG user simulation tells us what the user is most likely to do next, *if we were to stop generating now*.

We are most interested in the following user reactions:

1. `select`: the user chooses one of the presented items, e.g. *"Yes, I'll take that one."*. This reply type indicates that the Information Presentation was sufficient for the user to make a choice.
2. `addInfo`: The user provides more attributes, e.g. *"I want something cheap."*. This reply type indicates that the user has more specific requests, which s/he wants to specify after being presented with the current information.
3. `requestMoreInfo`: The user asks for more information, e.g. *"Can you recommend me one?", "What is the price range of the last item?"*. This reply type indicates that the system failed to present the information the user was looking for.
4. `askRepeat`: The user asks the system to repeat the same message again, e.g. *"Can you repeat?"*. This reply type indicates that the utterance was either too long or confusing for the user to remember, or the TTS quality was not good enough, or both.
5. `silence`: The user does not say anything. In this case it is up to the system to take initiative.
6. `hangup`: The user closes the interaction.

We build user simulations using n-gram models of system (s) and user (u) acts, as first introduced by (Eckert et al., 1997). In order to account for data sparsity, we apply different *discounting* ("smoothing") techniques including *back-off*, using the CMU Statistical Language Modelling toolkit (Clarkson and Rosenfeld, 1997). We construct a **bi-gram** model[5] for the users' reactions to the system's IP structure decisions ($P(a_{u,t}|IP_{s,t})$), and a **tri-gram** (i.e. IP structure + attribute choice) model for predicting user reactions to the system's combined IP structure and attribute selection decisions: $P(a_{u,t}|IP_{s,t}, attributes_{s,t})$.

[5]Where $a_{u,t}$ is the predicted next user action at time t, $IP_{s,t}$ was the system's Information Presentation action at t, and $attributes_{s,t}$ is the attributes selected by the system at t.

We evaluate the performance of these models by measuring dialogue similarity to the original data, based on the Kullback-Leibler (KL) divergence, as also used by, e.g. (Cuayáhuitl et al., 2005; Jung et al., 2009; Janarthanam and Lemon, 2009). We compare the raw probabilities as observed in the data with the probabilities generated by our n-gram models using different discounting techniques for each context, see table 2. All the models have a small divergence from the original data (especially the bi-gram model), suggesting that they are reasonable simulations for training and testing NLG policies.

The absolute discounting method for the bi-gram model is most dissimilar to the data, as is the WittenBell method for the tri-gram model, i.e. the models using these discounting methods have the highest KL score. The best performing methods (i.e. most similar to the original data), are linear discounting for the bi-gram model and GoodTuring for the tri-gram. We use the most similar user models for system training, and the most dissimilar user models for testing NLG policies, in order to test whether the learned policies are robust and adaptive to unseen dialogue contexts.

discounting method	bi-gram US	tri-gram US
WittenBell	0.086	*0.512*
GoodTuring	0.086	**0.163**
absolute	*0.091*	0.246
linear	**0.011**	0.276

Table 2: Kullback-Leibler divergence for the different User Simulations (US)

4.2 Database matches and "Focus of attention"

An important task of Information Presentation is to support the user in choosing between all the available items (and ultimately in selecting the most suitable one) by structuring the current information returned from the database, as explained in Section 1.1. We therefore model the user's "focus of attention" as a feature in our learning experiments. This feature reflects how the different IP strategies structure information with different numbers of attributes. We implement this shift of the user's focus analogously to discovering the user's goal in Dialogue Management: every time the predicted next user act is to add in-

formation (addInfo), we infer that the user is therefore only interested in a subset of the previously presented results and so the system will focus on this new subset of database items in the rest of the generated utterance. For example, the user's focus after the SUMMARY (with UM) in Table 1 is $DBhits = 10$, since the user is only interested in cheap, Indian places.

4.3 Data-driven Reward function

The reward/evaluation function is constructed from the WoZ data, using a stepwise linear regression, following the PARADISE framework (Walker et al., 2000). This model selects the features which significantly influenced the users' ratings for the NLG strategy in the WoZ questionnaire. We also assign a value to the user's reactions ($valueUserReaction$), similar to optimising task success for DM (Young et al., 2007). This reflects the fact that good IP strategies should help the user to select an item ($valueUserReaction = +100$) or provide more constraints addInfo ($valueUserReaction = \pm 0$), but the user should not do anything else ($valueUserReaction = -100$). The regression in equation 1 ($R^2 = .26$) indicates that users' ratings are influenced by higher level and lower level features: Users like to be focused on a small set of database hits (where $\#DBhits$ ranges over [1-100]), which will enable them to choose an item ($valueUserReaction$), while keeping the IP utterances short (where $\#sentence$ is in the range [2-18]):

$$\begin{aligned} Reward = (-1.2) \times \#DBhits \\ +(.121) \times valueUserReaction \\ -(1.43) \times \#sentence \end{aligned} \quad (1)$$

Note that the worst possible reward for an NLG move is therefore $(-1.20 \times 100) - (.121 \times 100) - (18 \times 1.43) = -157.84$. This is achieved by presenting 100 items to the user in 18 sentences[6], in such a way that the user ends the conversation unsuccessfully. The top possible reward is achieved in the rare cases where the system can immediately present 1 item to the user using just 2 sentences, and the user then selects that item, i.e. Reward = $-(1.20 \times 1) + (.121 \times 100) - (2 \times 1.43) = 8.06$

[6]Note that the maximum possible number of sentences generated by the realizer is 18 for the full IP sequence SUMMARY+COMPARE+RECOMMEND using all the attributes.

5 Reinforcement Learning experiments

We now formulate the problem as a Markov Decision Process (MDP), where states are NLG dialogue contexts and actions are NLG decisions. Each state-action pair is associated with a transition probability, which is the probability of moving from state s at time t to state s' at time $t+1$ after having performed action a when in state s. This transition probability is computed by the environment model (i.e. the user simulation and realiser), and explicitly captures the uncertainty in the generation environment. This is a major difference to other non-statistical planning approaches. Each transition is also associated with a reinforcement signal (or "reward") r_{t+1} describing how good the result of action a was when performed in state s. The aim of the MDP is to maximise long-term expected reward of its decisions, resulting in a *policy* which maps each possible state to an appropriate action in that state.

We treat IP as a hierarchical joint optimisation problem, where first one of the IP structures (1-3) is chosen and then the number of attributes is decided, as shown in Figure 4. At each generation step, the MDP can choose 1-5 attributes (e.g. cuisine, price range, location, food quality, and/or service quality). Generation stops as soon as the user is predicted to select an item, i.e. the IP task is successful. (Note that the same constraint is operational for the WoZ baseline.)

$$\left[\begin{array}{l} \text{ACTION:} \left[\begin{array}{l} \textit{IP:} \left\{\begin{array}{l}\text{SUMMARY}\\ \text{COMPARE}\\ \text{RECOMMEND}\end{array}\right\} \left\{\text{attr: } 1\text{-}5\right\} \end{array}\right] \\ \text{STATE:} \left[\begin{array}{l} \texttt{attributes:}\left\{\texttt{1-15}\right\} \\ \texttt{sentence:}\left\{\texttt{2-18}\right\} \\ \texttt{dbHitsFocus:}\left\{\texttt{1-100}\right\} \\ \texttt{userSelect:}\left\{\texttt{0,1}\right\} \\ \texttt{userAddInfo:}\left\{\texttt{0,1}\right\} \\ \texttt{userElse:}\left\{\texttt{0,1}\right\} \end{array}\right] \end{array}\right]$$

Figure 4: State-Action space for the RL-NLG problem

States are represented as sets of NLG dialogue context features. The state space comprises "lower-level" features about the realiser behaviour (two discrete features representing the number of attributes and sentences generated so far) and three binary features representing the user's predicted next action, as well as "high-level" features pro-

vided by the DM (e.g. current database hits in the user's focus (`dbHitsFocus`)). We trained the policy using the SHARSHA algorithm (Shapiro and Langley, 2002) with linear function approximation (Sutton and Barto, 1998), and the simulation environment described in Section 4. The policy was trained for 60,000 iterations.

5.1 Experimental Set-up

We compare the learned strategies against the *WoZ baseline* as described in Section 3.3. For attribute selection we choose a majority baseline (randomly choosing between 3 or 4 attributes) since the attribute selection models learned by Supervised Learning on the WoZ data didn't show significant improvements.

For training, we used the user simulation model most similar to the data, see Section 4.1. For testing, we test with the *different* user simulation model (the one which is most dissimilar to the data).

We first investigate how well IP structure (without attribute choice) can be learned in increasingly complex generation **scenarios**. A generation scenario is a combination of a particular kind of NLG realiser (template vs. stochastic) along with different levels of variation introduced by certain features of the dialogue context. In general, the stochastic realiser introduces more variation in lower level features than the template-based realiser. The Focus model introduces more variation with respect to #DBhits and #attributes as described in Section 4.2. We therefore investigate the following cases:

1.1. IP structure choice, Template realiser: Predicted next user action varies according to the bi-gram model ($P(a_{u,t}|IP_{s,t})$); Number of sentences and attributes per IP strategy is set by defaults, reflecting a template-based realiser.

1.2. IP structure choice, Stochastic realiser: IP structure where number of attributes per NLG turn is given at the beginning of each episode (e.g. set by the DM); Sentence generation according to the SPaRKy stochastic realiser model as described in Section 3.2.

We then investigate different scenarios for *jointly* optimising IP structure (IPS) and attribute selection (Attr) decisions.

2.1. IPS+Attr choice, Template realiser: Predicted next user action varies according to tri-gram ($P(a_{u,t}|IP_{s,t}, attributes_{s,t})$) model; Number of sentences per IP structure set to default.

2.2. IPS+Attr choice, Template realiser+Focus model: Tri-gram user simulation with Template realiser and Focus of attention model with respect to #DBhits and #attributes as described in section 4.2.

2.3. IPS+Attr choice, Stochastic realiser: Tri-gram user simulation with sentence/attribute relationship according to Stochastic realiser as described in Section 3.2.

2.4. IPS+Attr choice, Stochastic realiser+Focus: i.e. the full model = Predicted next user action varies according to tri-gram model+ Focus of attention model + Sentence/attribute relationship according to stochastic realiser.

5.2 Results

We compare the average final reward (see Equation 1) gained by the baseline against the trained RL policies in the different scenarios for each 1000 test runs, using a paired samples t-test. The results are shown in Table 3. In 5 out of 6 scenarios the RL policy significantly ($p < .001$) outperforms the supervised baseline. We also report on the percentage of the top possible reward gained by the individual policies, and the raw percentage improvement of the RL policy. Note that the best possible (100%) reward can only be gained in rare cases (see Section 4.3).

The learned RL policies show that lower level features are important in gaining significant improvements over the baseline. The more complex the scenario, the harder it is to gain higher rewards for the policies in general (as more variation is introduced), but the relative improvement in rewards also increases with complexity: the baseline does not adapt well to the variations in lower level features whereas RL learns to adapt to the more challenging scenarios. [7]

An overview of the range of different IP strategies learned for each setup can be found in Table 4. Note that these strategies are context-dependent: the learner chooses how to proceed dependent on

[7] Note, that the baseline does reasonably well in scenarios with variation introduced by only higher level features (e.g. scenario 2.2).

Scenario	Wizard Baseline average Reward	RL average Reward	RL % - Baseline % = % improvement
1.1	-15.82(±15.53)	-9.90***(±15.38)	**89.2%** - 85.6%= 3.6%
1.2	-19.83(±17.59)	-12.83***(±16.88)	**87.4%** - 83.2%= 4.2%
2.1	-12.53(±16.31)	-6.03***(±11.89)	**91.5%** - 87.6%= 3.9%
2.2	-14.15(±16.60)	-14.18(±18.04)	**86.6%** - 86.6%= 0.0%
2.3	-17.43(±15.87)	-9.66***(±14.44)	**89.3%** - 84.6%= 4.7%
2.4	-19.59(±17.75)	-12.78***(±15.83)	**87.4%** - 83.3%= 4.1%

Table 3: Test results for 1000 dialogues, where *** denotes that the RL policy is significantly ($p < .001$) better than the Baseline policy.

the features in the state space at each generation step.

Scenario	strategies learned
1.1	RECOMMEND COMPARE COMPARE+RECOMMEND SUMMARY SUMMARY+COMPARE SUMMARY+RECOMMEND SUMMARY+COMPARE+RECOMMEND.
1.2	RECOMMEND COMPARE COMPARE+RECOMMEND SUMMARY SUMMARY+COMPARE SUMMARY+RECOMMEND SUMMARY+COMPARE+RECOMMEND.
2.1	RECOMMEND(5) SUMMARY(2) SUMMARY(2)+COMPARE(4) SUMMARY(2)+COMPARE(1) SUMMARY(2)+COMPARE(4)+RECOMMEND(5) SUMMARY(2)+COMPARE(1)+RECOMMEND(5)
2.2	RECOMMEND(5) SUMMARY(4) SUMMARY(4)+RECOMMEND(5)
2.3	RECOMMEND(2) SUMMARY(1) SUMMARY(1)+COMPARE(4) SUMMARY(1)+COMPARE(1) SUMMARY(1)+COMPARE(4)+RECOMMEND(2)
2.4	RECOMMEND(2) SUMMARY(2) SUMMARY(2)+COMPARE(4) SUMMARY(2)+RECOMMEND(2) SUMMARY(2)+COMPARE(4)+RECOMMEND(2) SUMMARY(2)+COMPARE(1)+RECOMMEND(2)

Table 4: RL strategies learned for the different scenarios, where (n) denotes the number of attributes generated.

For example, the RL policy for scenario 1.1 learned to start with a SUMMARY if the initial number of items returned from the database is high (>30). It will then stop generating if the user is predicted to select an item. Otherwise, it continues with a RECOMMEND. If the number of database items is low, it will start with a COMPARE and then continue with a RECOMMEND, unless the user selects an item. Also see Table 4. Note that the WoZ strategy behaves as described in Figure 3.

In addition, the RL policy for scenario 1.2 learns to adapt to a more complex scenario: the number of attributes requested by the DM and produced by the stochastic sentence realiser. It learns to generate the whole sequence (SUMMARY+COMPARE+RECOMMEND) if #attributes is low (<3), because the overall generated utterance (final #sentences) is still relatively short. Otherwise the policy is similar to the one for scenario 1.1.

The RL policies for jointly optimising IP strategy and attribute selection learn to select the number of attributes according to the generation scenarios 2.1-2.4. For example, the RL policy learned for scenario 2.1 generates a RECOMMEND with 5 attributes if the database hits are low (<13). Otherwise, it will start with a SUMMARY using 2 attributes. If the user is predicted to narrow down his focus after the SUMMARY, the policy continues with a COMPARE using 1 attribute only, otherwise it helps the user by presenting 4 attributes. It then continues with RECOMMEND(5), and stops as soon as the user is predicted to select one item.

The learned policy for scenario 2.1 generates 5.85 attributes per NLG turn on average (i.e. the cumulative number of attributes generated in the whole NLG sequence, where the same attribute may be repeated within the sequence). This strategy primarily adapts to the variations from the user simulation (tri-gram model). For scenario 2.2 the average number of attributes is higher (7.15) since the number of attributes helps to narrow down the user's focus via the DBhits/attribute relationship specified in section 4.2. For scenario 2.3 fewer attributes are generated on average (3.14), since here the number of attributes influences the sentence realiser, i.e. fewer attributes results in fewer sentences, but does not impact the user's focus. In scenario 2.4 all the conditions mentioned above influence the learned policy. The average number of attributes selected is still low (3.19).

In comparison, the average (cumulative) num-

ber of attributes for the WoZ baseline is 7.10. The WoZ baseline generates all the possible IP structures (with 3 or 4 attributes) but is restricted to use only "high-level" features (see Figure 3). By beating this baseline we show the importance of the "lower-level" features. Nevertheless, this wizard policy achieves up to 87.6% of the possible reward on this task, and so can be considered a serious baseline against which to measure performance.

The only case (scenario 2.2) where RL does not improve significantly over the baseline is where lower level features do not play an important role for learning good strategies: scenario 2.2 is only sensitive to higher level features (DBhits).

6 Conclusion

We have presented a new data-driven method for Information Presentation (IP) in Spoken Dialogue Systems using a statistical optimisation framework for content structure planning and attribute selection. This work is the first to apply a data-driven optimisation method to the IP decision space, and to show the utility of both lower-level and higher-level features for this problem.

We collected data in a Wizard-of-Oz (WoZ) experiment and showed that human "wizards" mostly pay attention to 'high-level' features from Dialogue Management. The WoZ data was used to build statistical models of user reactions to IP strategies, and a data-driven reward function for Reinforcement Learning (RL). We show that lower level features significantly influence users' ratings of IP strategies. We compared a model of human behaviour (the 'human wizard baseline') against policies optimised using Reinforcement Learning, in a variety of scenarios. Our optimised policies significantly outperform the IP structuring and attribute selection present in the WoZ data, especially when performing in complex generation scenarios which require adaptation to, e.g. number of database results, utterance length, etc. While the human wizards were able to attain up to 87.6% of the possible reward on this task, the RL policies are significantly better in 5 out of 6 scenarios, gaining up to 91.5% of the total possible reward.

We have also shown that adding predictive "lower level" features, e.g. from the NLG realiser and a user reaction model, is important for learning optimal IP strategies according to user preferences. Future work could include the predicted TTS quality (Boidin et al., 2009) as a feature.

We are now working on testing the learned policies with real users, outside of laboratory conditions, using a restaurant-guide SDS, deployed as a VOIP service. Previous work in SDS has shown that results for Dialogue Management obtained with simulated users are able to transfer to evaluations with real users (Lemon et al., 2006).

This methodology provides new insights into the nature of the IP problem, which has previously been treated as a module following dialogue management with no access to lower-level context features. The data-driven planning method applied here promises a significant upgrade in the performance of generation modules, and thereby of Spoken Dialogue Systems in general.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216594 (CLASSiC project `www.classic-project.org`) and from the EPSRC, project no. EP/G069840/1.

References

Cedric Boidin, Verena Rieser, Lonneke van der Plas, Oliver Lemon, and Jonathan Chevelu. 2009. Predicting how it sounds: Re-ranking alternative inputs to TTS using latent variables (forthcoming). In *Proc. of Interspeech/ICSLP, Special Session on Machine Learning for Adaptivity in Spoken Dialogue Systems*.

Grace Chung. 2004. Developing a flexible spoken dialog system using simulation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

P.R. Clarkson and R. Rosenfeld. 1997. Statistical Language Modeling Using the CMU-Cambridge Toolkit. In *Proc. of ESCA Eurospeech*.

William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer dialogue simulation using hidden markov models. In *Proc. of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Vera Demberg and Johanna D. Moore. 2006. Information presentation in spoken dialogue systems. In *Proceedings of EACL*.

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Proc. of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*.

M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and S. Young. 2008. Training and Evaluation of the HIS POMDP Dialogue System in Noise. In *Proc. of SIGdial Workshop on Discourse and Dialogue*.

James Henderson and Oliver Lemon. 2008. Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management. In *Proc. of ACL*.

Srinivasan Janarthanam and Oliver Lemon. 2009. A Two-tier User Simulation Model for Reinforcement Learning of Adaptive Referring Expression Generation Policies. In *Proc. of SIGdial*.

Srini Janarthanam and Oliver Lemon. 2010. Learning to adapt to unknown users: Referring expression generation in spoken dialogue systems. In *Proceedings of ACL*.

Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer, Speech & Language*, 23:479–509.

Alexander Koller and Ronald Petrick. 2008. Experiences with planning for natural language generation. In *ICAPS*.

Oliver Lemon, Kallirroi Georgila, and James Henderson. 2006. Evaluating Effectiveness and Portability of Reinforcement Learned Dialogue Strategies with real users: the TALK TownInfo Evaluation. In *IEEE/ACL Spoken Language Technology*.

Oliver Lemon. 2008. Adaptive Natural Language Generation in Dialogue using Reinforcement Learning. In *Proceedings of SEMdial*.

Oliver Lemon. 2010. Learning what to say and how to say it: joint optimization of spoken dialogue management and Natural Language Generation. *Computer, Speech & Language*, to appear.

Xingkun Liu, Verena Rieser, and Oliver Lemon. 2009. A wizard-of-oz interface to study information presentation strategies for spoken dialogue systems. In *Proc. of the 1st International Workshop on Spoken Dialogue Systems*.

Crystal Nakatsu. 2008. Learning contrastive connectives in sentence realization ranking. In *Proc. of SIGdial Workshop on Discourse and Dialogue*.

Joseph Polifroni and Marilyn Walker. 2006. Learning database content for spoken dialogue system design. In *Proc. of the IEEE/ACL workshop on Spoken Language Technology (SLT)*.

Joseph Polifroni and Marilyn Walker. 2008. Intensional Summaries as Cooperative Responses in Dialogue Automation and Evaluation. In *Proceedings of ACL*.

Verena Rieser and Oliver Lemon. 2008. Learning Effective Multimodal Dialogue Strategies from Wizard-of-Oz data: Bootstrapping and Evaluation. In *Proc. of ACL*.

Verena Rieser and Oliver Lemon. 2009. Natural Language Generation as Planning Under Uncertainty for Spoken Dialogue Systems. In *Proc. of EACL*.

Verena Rieser, Xingkun Liu, and Oliver Lemon. 2009. Optimal Wizard NLG Behaviours in Context. Technical report, Deliverable 4.2, CLASSiC Project.

Dan Shapiro and P. Langley. 2002. Separating skills from preference: Using learning to program by reward. In *Proc. of the 19th International Conference on Machine Learning (ICML)*.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Association for Computational Linguistics*.

R. Sutton and A. Barto. 1998. *Reinforcement Learning*. MIT Press.

Kees van Deemter. 2009. What game theory can do for NLG: the case of vague language. In *12th European Workshop on Natural Language Generation (ENLG)*.

Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. 2000. Towards developing general models of usability with PARADISE. *Natural Language Engineering*, 6(3).

M. Walker, R. Passonneau, and J. Boland. 2001. Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456.

Steve Whittaker, Marilyn Walker, and Johanna Moore. 2002. Fish or Fowl: A Wizard of Oz evaluation of dialogue strategies in the restaurant domain. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*.

Andi Winterboer, Jiang Hu, Johanna D. Moore, and Clifford Nass. 2007. The influence of user tailoring and cognitive load on user performance in spoken dialogue systems. In *Proc. of the 10th International Conference of Spoken Language Processing (Interspeech/ICSLP)*.

SJ Young, J Schatzmann, K Weilhammer, and H Ye. 2007. The Hidden Information State Approach to Dialog Management. In *ICASSP 2007*.

Combining data and mathematical models of language change

Morgan Sonderegger
University of Chicago
Chicago, IL, USA.
morgan@cs.uchicago.edu

Partha Niyogi
University of Chicago
Chicago, IL, USA.
niyogi@cs.uchicago.edu

Abstract

English noun/verb (N/V) pairs (*contract*, *cement*) have undergone complex patterns of change between 3 stress patterns for several centuries. We describe a longitudinal dataset of N/V pair pronunciations, leading to a set of properties to be accounted for by any computational model. We analyze the dynamics of 5 dynamical systems models of linguistic populations, each derived from a model of learning by individuals. We compare each model's dynamics to a set of properties observed in the N/V data, and reason about how assumptions about individual learning affect population-level dynamics.

1 Introduction

The fascinating phenomena of language evolution and language change have inspired much work from computational perspectives in recent years. Research in this field considers populations of linguistic agents, and asks how the population dynamics are related to the behavior of individual agents. However, most such work makes little contact with empirical data (de Boer and Zuidema, 2009).[1] As pointed out by Choudhury (2007), most computational work on language change deals with data from cases of change either not at all, or at a relatively high level.[2]

Recent computational work has addressed "real world" data from change in several languages (Mitchener, 2005; Choudhury et al., 2006; Choudhury et al., 2007; Pearl and Weinberg, 2007; Daland et al., 2007; Landsbergen, 2009). In the same spirit, we use data from an ongoing stress shift in English noun/verb (N/V) pairs. Because stress has been listed in dictionaries for several centuries, we are able to trace stress longitudinally and at the level of individual words, and observe dynamics significantly more complicated than in changes previously considered in the computational literature. In §2, we summarize aspects of the dynamics to be accounted for by any computational model of the stress shift. We also discuss proposed sources of these dynamics from the literature, based on experimental work by psychologists and linguists.

In §3–4, we develop models in the mathematical framework of dynamical systems (DS), which over the past 15 years has been used to model the interaction between language learning and language change in a variety of settings (Niyogi and Berwick, 1995; Niyogi and Berwick, 1996; Niyogi, 2006; Komarova et al., 2001; Yang, 2001; Yang, 2002; Mitchener, 2005; Pearl and Weinberg, 2007).

We interpret 6 aspects of the N/V stress dynamics in DS terms; this gives a set of 6 desired properties to which any DS model's dynamics can be compared. We consider 5 models of language learning by individuals, based on the experimental findings relevant to the N/V stress shift, and evaluate the population-level dynamics of the dynamical system model resulting from each against the set of desired properties. We are thus able to reason about which theories of the source of language change — considered as hypotheses about how individuals learn — lead to the population-level patterns observed in change.

2 Data: English N/V pairs

The data considered here are the stress patterns of English homographic, disyllabic noun/verb pairs (Table 1); we refer to these throughout as "N/V pairs". Each of the N and V forms of a pair can have initial ($\acute{\sigma}\sigma$: *cónvict*, n.) or final ($\sigma\acute{\sigma}$: *convíct*,

[1] However, among language evolution researchers there has been significant recent interest in behavioral experiments, using the "iterated learning" paradigm (Griffiths and Kalish, 2007; Kalish et al., 2007; Kirby et al., 2008).

[2] We do not review the literature on computational studies of change due to space constraints; see (Baker, 2008; Wang et al., 2005; Niyogi, 2006) for reviews.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1019–1029,
Uppsala, Sweden, 11-16 July 2010.

	N	V	
$\{1,1\}$	$\acute{\sigma}\sigma$	$\acute{\sigma}\sigma$	(exile, anchor, fracture)
$\{1,2\}$	$\acute{\sigma}\sigma$	$\sigma\acute{\sigma}$	(consort, protest, refuse)
$\{2,2\}$	$\sigma\acute{\sigma}$	$\sigma\acute{\sigma}$	(cement, police, review)

Table 1: Attested N/V pair stress patterns.

v.) stress. We use the notation {Nstress,Vstress} to denote the stress of an N/V pair, with 1=$\acute{\sigma}\sigma$, 2=$\sigma\acute{\sigma}$. Of the four logically possible stress patterns, all current N/V pairs follow one of the 3 patterns shown in Table 1: {1,1}, {1,2}, {2,2}.[3] No pair follows the fourth possible pattern, {2,1}.

N/V pairs have been undergoing variation and change between these 3 patterns since Middle English (ME, c. 1066-1470), especially change to {1,2}. The vast majority of stress shifts occurred after 1570 (Minkova, 1997), when the first dictionary listing English word stresses was published (Levens, 1570). Many dictionaries from the 17th century on list word stresses, making it possible to trace change in the stress of individual N/V pairs in considerable detail.

2.1 Dynamics

Expanding on dictionary pronunciation data collected by Sherman (1975) for the period 1570–1800, we have collected a corpus of pronunciations of 149 N/V pairs, as listed in 62 British dictionaries, published 1570–2007. Variation and change in N/V pair stress can be visualized by plotting *stress trajectories*: the moving average of N and V stress vs. time for a given pair. Some examples are shown in Fig. 1. The corpus is described in detail in (Sonderegger and Niyogi, 2010); here we summarize the relevant facts to be accounted for in a computational model.[4]

Change Four types of clear-cut change between the three stress patterns are observed:

{2,2}→{1,2} (Fig.1(a)) {1,2}→{1,1}
{1,1}→{1,2} (Fig. 1(b)) {1,2}→{2,2}

However, change to {1,2} is much more common than change from {1,2}; in particular, {2,2}→{1,2} is the most common change. When change occurs, it is often fairly sudden, as in Figs. 1(a), 1(b). Finally, change never occurs *directly* between {1,1} and {2,2}.

Stability Previous work on stress in N/V pairs (Sherman, 1975; Phillips, 1984) has emphasized change, in particular {2,2}→{1,2} (the most common change). However, an important aspect of the diachronic dynamics of N/V pairs is stability: most N/V pairs do *not* show variation or change.

The 149 N/V pairs, used both in our corpus and in previous work, were chosen by Sherman (1975) as those most likely to have undergone change, and thus are not suitable for studying how stable the three attested stress patterns are. In a *random* sample of N/V pairs (not the set of 149) in use over a fixed time period (1700–2007), we find that only 12% have shown variation or change in stress (Sonderegger and Niyogi, 2010). Most pairs maintain the {1,1}, {2,2}, or {1,2} stress pattern for hundreds of years. A model of the diachronic dynamics of N/V pair stress must explain how it can be the case both that some pairs show variation and change, and that many do not.

Variation N/V pair stress patterns show both synchronic and diachronic variation.

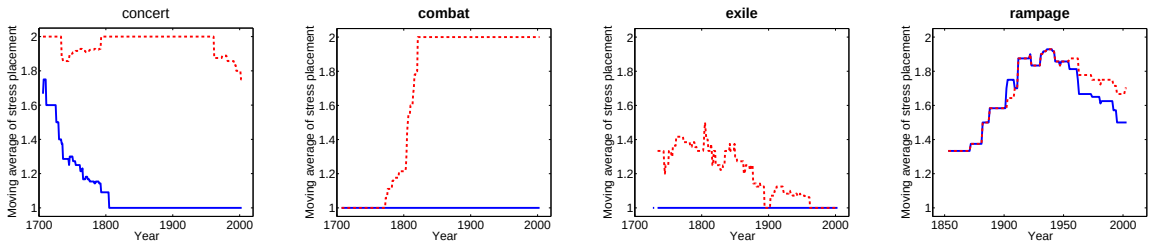
Synchronically, there is variation at the population level in the stress of some N/V pairs at any given time; this is reflected by the inclusion of more than one pronunciation for some N/V pairs in many dictionaries. An important question for modeling is whether there is variation within *individual* speakers. We show in (Sonderegger and Niyogi, 2010) that there is, for present-day American English speakers, using a corpus of radio speech. For several N/V pairs which have currently variable pronunciation, 1/3 of speakers show variation in the stress of the N form. Metrical evidence from poetry suggests that individual variation also existed in the past; the best evidence is for Shakespeare, who shows variation in the stress of over 20 N/V pairs (Kökeritz, 1953).

Diachronically, a relevant question for modeling is whether all variation is short-lived, or whether *stable variation* is possible. A particular type of stable variation is in fact observed relatively often in the corpus: *either* the N or V form stably vary (Fig. 1(c)), but not both at once. Stable variation where both N and V forms vary almost never occurs (Fig. 1(d)).

Frequency dependence Phillips (1984) hypoth-

[3] However, as variation and change in N/V pair stress is ongoing, a few pairs (e.g. *perfume*) currently have variable stress. By "stress", we always mean "primary stress". All present-day pronunciations are for British English, from CELEX (Baayen et al., 1996).

[4] The corpus is available on the first author's home page (currently, `people.cs.uchicago.edu/~morgan`).



(a) concert (b) combat (c) exile (d) rampage

Figure 1: Example N/V pair stress trajectories. Moving averages (60-year window) of stress placement (1=$\acute{\sigma}\sigma$, 2=$\sigma\acute{\sigma}$). Solid lines=nouns, dashed lines=verbs.

esizes that N/V pairs with lower frequencies (summed N+V word frequencies) are more likely to change to {1,2}. Sonderegger (2010) shows that this is the case for the most common change, {2,2}→{1,2}: among N/V pairs which were {2,2} in 1700 and are either {2,2} or {1,2} today, those which have undergone change have significantly lower frequencies, on average, than those which have not. In (Sonderegger and Niyogi, 2010), we give preliminary evidence from real-time frequency trajectories (for <10 N/V pairs) that it is not lower frequency *per se* which triggers change to {1,2}, but *falling* frequency. For example, change in *combat* from {1,1}→{1,2} around 1800 (Fig. 1(b)) coincides with falling word frequency from 1775–present.

2.2 Sources of change

The most salient facts about English N/V pair stress are that (a) change is most often to {1,2} (b) the {2,1} pattern never occurs. We summarize two types of explanation for these facts from the experimental literature, each of which exemplifies a commonly-proposed type of explanation for phonological change. In both cases, there is experimental evidence for biases in present-day English speakers reflecting (a–b). We assume that these biases have been active over the course of the N/V stress shift, and can thus be seen as possible sources of the diachronic dynamics of N/V pairs.[5]

Analogy/Lexicon In historical linguistics, analogical changes are those which make "...related forms more similar to each other in their phonetic (and morphological) structure" (Hock, 1991).[6] Proposed causes for analogical change thus often involve a speaker's production and perception of a form being influenced by similar forms in their lexicon.

The English lexicon shows a broad tendency, which we call *Ross' generalization*, which could be argued to be driving analogical change to {1,2}, and acting against the unobserved stress pattern {2,1}: "primary stress in English nouns is farther to the left than primary stress in English verbs" (Ross, 1973). Change to {1,2} could be seen as motivated by Ross' generalization, and {2,1} made impossible by it.

The argument is lent plausibility by experimental evidence that Ross' Generalization is reflected in production and perception. English listeners strongly prefer the typical stress pattern (N=$\acute{\sigma}\sigma$ or V=$\sigma\acute{\sigma}$) in novel English disyllables (Guion et al., 2003), and process atypical disyllables (N=$\sigma\acute{\sigma}$ or V=$\acute{\sigma}\sigma$) more slowly than typical ones (Arciuli and Cupples, 2003).

Mistransmission An influential line of research holds that many phonological changes are based in asymmetric transmission errors: because of articulatory or perceptual factors, listeners systematically mishear some sound α as β, but rarely mishear β as α.[7] We call such effects *mistransmission*. Asymmetric mistransmission (by individu-

[5] This type of assumption is necessary for any hypothesis about the sources of a completed or ongoing change, based on present-day experimental evidence, and is thus common in the literature. In the case of N/V pairs, it is implicitly made in Kelly's (1988 *et seq*) account, discussed below. Both biases discussed here stem from facts about English (Ross' Generalization; rhythmic context) that we believe have not changed over the time period considered here (≈1600–present), based on general accounts of English historical phonology during this period (Lass, 1992; MacMahon, 1998). We leave more careful verification of this claim to future work.

[6] "Forms" here means any linguistic unit; e.g. sounds, words, or paradigms, such as an N/V pair's stress pattern.

[7] A standard example is final obstruent devoicing, a common change cross-linguistically. There are several articulatory and perceptual reasons why final voiced obstruents could be heard as unvoiced, but no motivation for the reverse process (final unvoiced obstruents heard as voiced) (Blevins, 2006).

als) is argued to be a necessary condition for the change $\alpha \rightarrow \beta$ at the population level, and an explanation for why the change $\alpha \rightarrow \beta$ is common, while the change $\beta \rightarrow \alpha$ is rarely (or never) observed. Mistransmission-based explanations were pioneered by Ohala (1981, *et seq.*), and are the subject of much recent work (reviewed by Hansson, 2008)

For English N/V pairs, M. Kelly and collaborators have shown mistransmission effects which they propose are responsible for the directionality of the most common type of N/V pair stress shifts ($\{1,1\}$, $\{2,2\} \rightarrow \{1,2\}$), based on "rhythmic context" (Kelly, 1988; Kelly and Bock, 1988; Kelly, 1989). Word stress is misperceived more often as initial in "trochaic-biasing" contexts, where the preceding syllable is weak or the following syllable is heavy; and more often as final in analogously "iambic-biasing" contexts. Nouns occur more frequently in trochaic contexts, and verbs more frequently in iambic contexts; there is thus pressure for the V forms of $\{1,1\}$ pairs to be misperceived as $\sigma\acute{\sigma}$, and for the N forms of $\{2,2\}$ pairs to be misperceived as $\acute{\sigma}\sigma$.

3 Modeling preliminaries

We first describe assumptions and notation for models developed below (§4).

Because of the evidence for within-speaker variation in N/V pair stress (§2.1), in all models described below, we assume that what is learned for a given N/V pair are the *probabilities* of using the $\sigma\acute{\sigma}$ form for the N and V forms.

We also make several simplifying assumptions. There are discrete generations G_t, and learners in G_t learn from G_{t-1}. Each example a learner in G_t hears is equally likely to come from any member of G_{t-1}. Each learner receives an identical number of examples, and each generation has infinitely many members.

These are idealizations, adopted here to keep models simple enough to analyze; the effects of relaxing some of these assumptions have been explored by Niyogi (2006) and Sonderegger (2009). The infinite-population assumption in particular makes the dynamics fully deterministic; this rules out the possibility of change due to *drift* (or *sample variation*), where a form disappears from the population because no examples of it are encountered by learners in G_t in the input from G_{t-1}.

Notation For a fixed N/V pair, a learner in G_t hears N_1 examples of the N form, of which k_1^t are $\sigma\acute{\sigma}$ and (N_1-k_1^t) are $\acute{\sigma}\sigma$; N_2 and k_2^t are similarly defined for V examples. Each example is sampled i.i.d. from a random member of G_{t-1}. The N_i are *fixed* (each learner hears the same number of examples), while the k_i^t are random variables (over learners in G_t). Each learner applies an algorithm $\mathcal{A}$ to the N_1+N_2 examples to learn $\hat{\alpha}_t, \hat{\beta}_t \in [0,1]$, the probabilities of *producing* N and V examples as $\sigma\acute{\sigma}$. α_t, β_t are the expectation of $\hat{\alpha}_t$ and $\hat{\beta}_t$ over members of G_t: $\alpha_t = E(\hat{\alpha}_t)$, $\beta_t = E(\hat{\beta}_t)$. $\hat{\alpha}_t$ and $\hat{\beta}_t$ are thus random variables (over learners in G_t), while $\alpha_t, \beta_t \in [0,1]$ are numbers.

Because learners in G_t draw examples at random from members of G_{t-1}, the distributions of $\hat{\alpha}_t$ and $\hat{\beta}_t$ are determined by $(\alpha_{t-1}, \beta_{t-1})$. (α_t, β_t), the expectations of $\hat{\alpha}_t$ and $\hat{\beta}_t$, are thus determined by $(\alpha_{t-1}, \beta_{t-1})$ via an *iterated map* f:

$$f : [0,1]^2 \rightarrow [0,1]^2, \quad f(\alpha_t, \beta_t) = (\alpha_{t+1}, \beta_{t+1}).$$

3.1 Dynamical systems

We develop and analyze models of populations of language learners in the mathematical framework of (discrete) dynamical systems (DS) (Niyogi and Berwick, 1995; Niyogi, 2006). This setting allows us to determine the diachronic, population-level consequences of assumptions about the learning algorithm used by individuals, as well as assumptions about population structure or the input they receive.

Because it is in general impossible to solve a given iterated map as a function of t, the dynamical systems viewpoint is to understand its long-term behavior by finding its fixed points and *bifurcations*: changes in the number and stability of fixed points as system parameters vary.

Briefly, α_* is a *fixed point* (FP) of f if $f(\alpha_*) = \alpha_*$; it is *stable* if $\lim_{t \rightarrow \infty} \alpha_t = \alpha_*$ for α_0 sufficiently near α_*, and *unstable* otherwise; these are also called *stable states* and *unstable states*. Intuitively, α_* is stable iff the system is stable under small perturbations from α_*.[8]

In the context of a linguistic population, change from state α (100% of the population uses $\{1,1\}$) to state β (100% of the population uses $\{1,2\}$) corresponds to a bifurcation, where some system parameter (N) passes a critical value (N_0). For

[8] See (Strogatz, 1994; Hirsch et al., 2004) for introductions to dynamical systems in general, and (Niyogi, 2006) for the type of models considered here.

$N<N_0$, α is stable. For $N>N_0$, α is unstable, and β is stable; this triggers change from α to β.

3.2 DS interpretation of observed dynamics

Below, we describe 5 DS models of linguistic populations. To interpret whether each model has properties consistent with the N/V dataset, we translate the observations about the dynamics of N/V stress made above (§2.1) into DS terms. This gives a list of desired properties against which to evaluate the properties of each model.

1. $^*\{2,1\}$: $\{2,1\}$ is not a stable state.
2. *Stability of* $\{1,1\}$, $\{1,2\}$, $\{2,2\}$: These stress patterns correspond to stable states (for some system parameter values).
3. *Observed stable variation*: Stable states are possible (for some system parameter values) corresponding to variation in the N or V form, but not both.
4. *Sudden change*: Change from one stress pattern to another corresponds to a bifurcation, where the fixed point corresponding to the old stress pattern becomes unstable.
5. *Observed changes*: There are bifurcations corresponding to each of the four observed changes ($\{1,1\} \rightleftharpoons \{1,2\}$, $\{2,2\} \rightleftharpoons \{1,2\}$).
6. *Observed frequency dependence*: Change to $\{1,2\}$ corresponds to a bifurcation in frequency (N), where $\{2,2\}$ or $\{1,1\}$ loses stability as N is decreased.

4 Models

We now describe 5 DS models, each corresponding to a learning algorithm $\mathcal{A}$ used by individual language learners. Each $\mathcal{A}$ leads to an iterated map, $f(\alpha_t, \beta_t) = (\alpha_{t+1}, \beta_{t+1})$, which describes the state of the population of learners over successive generations. We give these evolution equations for each model, then discuss their dynamics, i.e. bifurcation structure. Each model's dynamics are evaluated with respect to the set of desired properties corresponding to patterns observed in the N/V data. Derivations have been mostly omitted for reasons of space, but are given in (Sonderegger, 2009).

The models differ along two dimensions, corresponding to assumptions about the learning algorithm ($\mathcal{A}$): whether or not it is assumed that the stress of examples is possibly mistransmitted (Models 1, 3, 5), and how the N and V probabilities acquired by a given learner are *coupled*. In Model 1 there is no coupling ($\hat{\alpha}_t$ and $\hat{\beta}_t$ learned independently), in Models 2–3 coupling takes the form of a hard constraint corresponding to Ross' generalization, and in Models 4–5 different stress patterns have different prior probabilities.[9]

4.1 Model 1: Mistransmission

Motivated by the evidence for asymmetric misperception of N/V pair stress (§2.2), suppose the stress of N=$\sigma\acute{\sigma}$ and V=$\acute{\sigma}\sigma$ examples may be misperceived (as N=$\acute{\sigma}\sigma$ and V=$\sigma\acute{\sigma}$), with *mistransmission probabilities* p and q.

Learners are assumed to simply probability match: $\hat{\alpha}_t = k_1^t/N_1$, $\hat{\beta}_t = k_2^t/N_2$, where k_1^t is the number of N and V examples *heard* as $\sigma\acute{\sigma}$ (etc.) The probabilities $p_{N,t}$ & $p_{V,t}$ of hearing an N or V example as final stressed at t are then

$$p_{N,t} = \alpha_{t-1}(1-p),\ p_{V,t} = \beta_{t-1} + (1-\beta_{t-1})q \quad (1)$$

k_1^t and k_2^t are binomially-distributed:

$$P_B(k_1^t, k_2^t) \equiv \binom{N_1}{k_1^t} {p_{N,t}}^{k_1^t} (1-p_{N,t})^{N_1-k_1^t} \times \binom{N_2}{k_2^t} {p_{V,t}}^{k_2^t} (1-p_{V,t})^{N_2-k_2^t} \quad (2)$$

α_t and β_t, the probability that a random member of G_t produces N and V examples as $\sigma\acute{\sigma}$, are the ensemble averages of $\hat{\alpha}_t$ and $\hat{\beta}_t$ over all members of G_t. Because we have assumed infinitely many learners per generation, α_t=$E(\hat{\alpha}_t)$ and β_t=$E(\hat{\beta}_t)$. Using (1), and the formula for the expectation of a binomially-distributed random variable:

$$\alpha_t = \alpha_{t-1}(1-p) \quad (3)$$
$$\beta_t = \beta_{t-1} + (1-\beta_{t-1})q \quad (4)$$

these are the *evolution equations* for Model 1. Due to space constraints we do not give the (more lengthy) derivations of the evolution equations in Models 2–5.

Dynamics There is a single, stable fixed point of evolution equations (3–4): $(\alpha_*, \beta_*) = (0, 1)$, corresponding to the stress pattern $\{1,2\}$. This model thus shows none of the desired properties discussed in §3.2, except that $\{1,2\}$ corresponds to a stable state.

[9] The sixth possible model (no coupling, no mistransmission) is a special case of Model 1, resulting in the identity map: $\alpha_{t+1} = \alpha_t$, $\beta_{t+1} = \beta_t$.

4.2 Model 2: Coupling by constraint

Motivated by the evidence for English speakers' productive knowledge of Ross' Generalization (§2.2), we consider a second learning model in which the learner attempts to probability match as above, but the $(\hat{\alpha}_t, \hat{\beta}_t)$ learned must satisfy the constraint that $\sigma\acute{\sigma}$ stress be more probable in the V form than in the N form.

Formally, the learner chooses $(\hat{\alpha}_t, \hat{\beta}_t)$ satisfying a quadratic optimization problem:

$$\text{minimize } [(\alpha - \frac{k_1^t}{N_1})^2 + (\beta - \frac{k_2^t}{N_2})^2] \text{ s.t. } \alpha \leq \beta$$

This corresponds to the following algorithm, $\mathcal{A}_2$:

1. If $\frac{k_1^t}{N_1} < \frac{k_2^t}{N_2}$, set $\hat{\alpha}_t = \frac{k_1^t}{N_1}$, $\hat{\beta}_t = \frac{k_2^t}{N_2}$.
2. Otherwise, set $\hat{\alpha}_t = \hat{\beta}_t = \frac{1}{2}(\frac{k_1^t}{N_1} + \frac{k_2^t}{N_2})$

The resulting evolution equations can be shown to be

$$\alpha_{t+1} = \alpha_t + \frac{A}{2}, \quad \beta_{t+1} = \beta_t - \frac{A}{2} \tag{5}$$

where $A = \sum_{\frac{k_1}{N_1} > \frac{k_2}{N_2}} P_B(k_1^t, k_2^t)(\frac{k_1^t}{N_1} - \frac{k_2^t}{N_2})$.

Dynamics Adding the equations in (5) gives that the (α_t, β_t) trajectories are lines of constant $\alpha_t + \beta_t$ (Fig. 2). All $(0, x)$ and $(x, 1)$ ($x \in [0, 1]$) are stable fixed points. This model thus has stable FPs corresponding to {1,1}, {1,2}, and {2,2}, does not have {2,1} as a stable FP (by construction), and allows for stable variation in exactly one of N or V. It does not have bifurcations, or the observed patterns of change and frequency dependence.

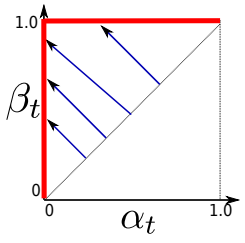


Figure 2: Dynamics of Model 2

4.3 Model 3: Coupling by constraint, with mistransmission

We now assume that each example is subject to mistransmission, as in Model 1; the learner then applies $\mathcal{A}_2$ to the *heard* examples. The evolution equations are thus the same as in (5), but with α_{t-1} and β_{t-1} changed to $p_{N,t}$, $p_{V,t}$ (Eqn. 1).

Dynamics There is a single, stable fixed point, corresponding to stable variation in both N and V. This model thus shows none of the desired properties, except that {2,1} is not a stable FP (by construction).

4.4 Model 4: Coupling by priors

The type of coupling assume in Models 2–3 — a constraint on the relative probability of $\sigma\acute{\sigma}$ stress for N and V forms — has the drawback that there is no way for the rest of the lexicon to affect a pair's N and V stress probabilities: there can be no influence of the stress of other N/V pairs, or in the lexicon as a whole, on the N/V pair being learned. Models 4–5 allow such influence by formalizing a simple intuitive explanation for the lack of $\{2, 1\}$ N/V pairs: learners cannot hypothesize a $\{2, 1\}$ pair because there is no support for this pattern in their lexicons.

We now assume that learners compute the probabilities of each possible N/V pair *stress pattern*, rather than separate probabilities for the N and V forms. We assume that learners keep two sets of probabilities (for $\{1, 1\}$, $\{1, 2\}$, $\{2, 1\}$, $\{2, 2\}$):

1. *Learned probabilities*:
   $\vec{P}$=$(P_{11}, P_{12}, P_{22}, P_{21})$, where

$$P_{11} = \frac{N_1 - k_1^t}{N_1}\frac{N_2 - k_2^t}{N_2}, \quad P_{12} = \frac{N_1 - k_1^t}{N_1}\frac{k_2^t}{N_2}$$
$$P_{22} = \frac{k_1^t}{N_1}\frac{k_2^t}{N_2}, \quad P_{21} = \frac{k_1^t}{N_1}\frac{N_2 - k_2^t}{N_2}$$

2. *Prior probabilities*: $\vec{\lambda} = (\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22})$, based on the support for each stress pattern in the lexicon.

The learner then produces N forms as follows:

1. Pick a pattern $\{n_1, v_1\}$ according to $\vec{P}$.
2. Pick a pattern $\{n_2, v_2\}$ according to $\vec{\lambda}$
3. Repeat 1–2 until n_1=n_2, then produce N=n_1.

V forms are produced similarly, but checking whether $v_1 = v_2$ at step 3. Learners' production of an N/V pair is thus influenced by both their learning experience (for the particular N/V pair) and by how much support exists in their lexicon for the different stress patterns.

We leave the exact interpretation of the λ_{ij} ambiguous; they could be the percentage of N/V pairs already learned which follow each stress pattern, for example. Motivated by the absence of {2,1} N/V pairs in English, we assume that $\lambda_{21} = 0$.

By following the production algorithm above, the learner's probabilities of producing N and V forms as $\sigma\acute{\sigma}$ are:

$$\hat{\alpha}_t = \tilde{\alpha}(k_1^t, k_2^t) = \frac{\lambda_{22}P_{22}}{\lambda_{11}P_{11} + \lambda_{12}P_{12} + \lambda_{22}P_{22}} \quad (6)$$

$$\hat{\beta}_t = \tilde{\beta}(k_1^t, k_2^t) = \frac{\lambda_{12}P_{12} + \lambda_{22}P_{22}}{\lambda_{11}P_{11} + \lambda_{12}P_{12} + \lambda_{22}P_{22}} \quad (7)$$

Eqns. 6–7 are undefined when (k_1^t, k_2^t)=$(N_1, 0)$; in this case we set $\tilde{\alpha}(N_1, 0) = \lambda_{22}$ and $\tilde{\beta}(N_1, 0) = \lambda_{12} + \lambda_{22}$.

The evolution equations are then

$$\alpha_t = E(\hat{\alpha}_t) = \sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2} P_B(k_1, k_2)\tilde{\alpha}(k_1, k_2) \quad (8)$$

$$\beta_t = E(\hat{\beta}_t) = \sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2} P_B(k_1, k_2)\tilde{\beta}(k_1, k_2) \quad (9)$$

Dynamics The fixed points of (8–9) are $(0,0)$, $(0,1)$, and $(1,1)$; their stabilities depend on N_1, N_2, and $\vec{\lambda}$. Define

$$R = \left(\frac{N_2}{1 + (N_2 - 1)\frac{\lambda_{12}}{\lambda_{11}}}\right)\left(\frac{N_1}{1 + (N_1 - 1)\frac{\lambda_{12}}{\lambda_{22}}}\right) \quad (10)$$

There are 6 regions of parameter space in which different FPs are stable:

1. $\lambda_{11}, \lambda_{22} < \lambda_{12}$: $(0,1)$ stable
2. $\lambda_{22} > \lambda_{12}$, $R < 1$: $(0,1)$, $(1,1)$ stable
3. $\lambda_{11} < \lambda_{12} < \lambda_{22}$, $R > 1$: $(1,1)$ stable
4. $\lambda_{11}, \lambda_{22} > \lambda_{12}$: $(0,0)$, $(1,1)$ stable
5. $\lambda_{22} < \lambda_{12} < \lambda_{11}$, $R > 1$: $(0,0)$ stable
6. $\lambda_{11} > \lambda_{12}$, $R < 1$: $(0,0)$, $(0,1)$ stable

The parameter space is split into these regimes by three hyperplanes: λ_{11}=λ_{12}, λ_{22}=λ_{12}, and R=1. Given that λ_{21}=0, $\lambda_{12} = 1 - \lambda_{11} - \lambda_{22}$, and the parameter space is 4-dimensional: $(\lambda_{11}, \lambda_{22}, N_1, N_2)$. Fig. 3 shows An example phase diagram in $(\lambda_{11}, \lambda_2)$, with N_1 and N_2 fixed.

The bifurcation structure implies all 6 possible changes between the three FPs ($\{1,1\}$⇌$\{1,2\}$, $\{1,2\}$⇌$\{2,2\}$, $\{2,2\}$⇌$\{1,2\}$). For example, suppose the system is at stable FP $(1,1)$ (corresponding to $\{2,2\}$) in region 2. As λ_{22} is decreased, we move into region 1, $(1,1)$ becomes unstable, and the system shifts to stable FP $(0,1)$. This transition corresponds to change from $\{2,2\}$ to $\{1,2\}$.

Note that change to $\{1,2\}$ entails crossing the hyperplanes λ_{12}=λ_{22} and λ_{12}=λ_{11}. These hyperplanes do not change as N_1 and N_2 vary, so change to $\{1,2\}$ is *not* frequency-dependent. However, change from $\{1,2\}$ entails crossing the hyperplane R=1, which does change as N_1 and N_2 vary (Eqn. 10), so change from $\{1,2\}$ is frequency-dependent. Thus, although there is frequency dependence in this model, it is not as observed in the diachronic data, where change *to* $\{1,2\}$ is frequency-dependent.

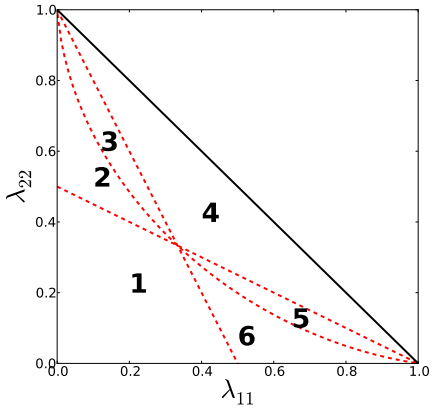


Figure 3: Example phase diagram in $(\lambda_{11}, \lambda_{22})$ for Model 4, with $N_1 = 5$, $N_2 = 10$. Numbers are regions of parameter space (see text).

Finally, no stable variation is possible: in every stable state, all members of the population categorically use a single stress pattern. $\{2,1\}$ is never a stable FP, by construction.

4.5 Model 5: Coupling by priors, with mistransmission

We now suppose that each example from a learner's data is possibly mistransmitted, as in Model 1; the learner then applies the algorithm from Model 4 to the *heard* examples (instead of using k_1^t, k_2^t) . The evolution equations are thus the same as (8–9), but with α_{t-1} and β_{t-1} changed to $p_{N,t}$, $p_{V,t}$ (Eqn. 1).

Dynamics $(0,1)$ is always a fixed point. For some regions of parameter space, there can be one fixed point of the form $(\kappa, 1)$, as well as one fixed point of the form $(0, \gamma)$, where $\kappa, \gamma \in (0,1)$. Define $R' = (1-p)(1-q)R$, $\lambda'_{12} = \lambda_{12}$, and

$$\lambda'_{11} = \lambda_{11}(1 - q\frac{N_2}{N_2 - 1}), \quad \lambda'_{22} = \lambda_{22}(1 - p\frac{N_1}{N_1 - 1})$$

There are 6 regions of parameter space corresponding to different stable FPs, identical to the 6 regions in Model 4, with the following substitu-

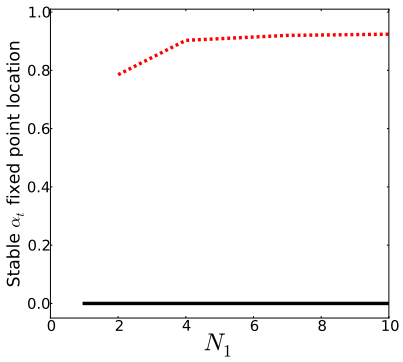


Figure 4: Example of falling N_1 triggering change from $(1,1)$ to $(0,1)$ for Model 5. Dashed line = stable FP of the form $(\gamma,1)$, solid line = stable FP $(0,1)$. For $N_1 > 4$, there is a stable FP near $(1,1)$. For $N_1 < 2$, $(0,1)$ is the only stable FP. $\lambda_{22} = 0.58$, $\lambda_{12} = 0.4$, $N_2 = 10$, $p = q = 0.05$.

tions made: $R \to R'$, $\lambda_{ij} \to \lambda'_{ij}$, $(0,0) \to (0,\kappa)$, $(1,1) \to (\gamma,1)$.

The parameter space is again split into these regions by three hyperplanes: λ'_{11}=λ'_{12}, λ'_{22}=λ'_{12}, and R'=1. As in Model 4, the bifurcation structure implies all 6 possible changes between the three FPs. However, change to {1,2} entails crossing the hyperplanes λ'_{11}=λ'_{12} and λ'_{2}=λ'_{12}, and is thus now frequency dependent.

In particular, consider a system at a stable FP $(\gamma,1)$, for some N/V pair. This FP becomes unstable if λ'_{22} becomes smaller than λ'_{12}. Assuming that the λ_{ij} are fixed, this occurs only if N_1 falls below a critical value, $N_1^* = (1 - \frac{\lambda_{22}}{\lambda_{12}}(1-p))^{-1}$; the system would then transition to $(0,1)$, the only stable state. By a similar argument, falling frequency can lead to change from $(0,\kappa)$ to $(0,1)$. Falling frequency can thus cause change to {1,2} in this model, as seen in the N/V data; Fig. 4 shows an example.

Unlike in Model 4, stable variation of the type seen in the N/V stress trajectories — one of N or V stably varying, but not both — is possible for some parameter values. $(0,0)$ and $(1,1)$ (corresponding to {1,1} and {2,2}) are technically never possible, but effectively occur for FPs of the form $(\kappa,0)$ and $(\gamma,1)$ when κ or γ are small. {2,1} is never a stable FP, by construction.

This model thus arguably shows all of the desired properties seen in the N/V data.

Property	Model				
	1	2	3	4	5
*{2,1}	✓	✓	✓	✓	✓
{1,1}, {1,2}, {2,2}	✗	✓	✗	✓	✓
Obs. stable variation	✗	✓	✗	✗	✓
Sudden change	✗	✗	✗	✓	✓
Observed changes	✗	✗	✗	✓	✓
Obs. freq. depend.	✗	✗	✗	✗	✓

Table 2: Summary of model properties

4.6 Models summary, observations

Table 2 lists which of Models 1–5 show each of the desired properties (from §3.2), corresponding to aspects of the observed diachronic dynamics of N/V pair stress.

Based on this set of models, we are able to make some observations about the effect of different assumptions about learning by individuals on population-level dynamics. Models including asymmetric mistransmission (1, 3, 5) generally do not lead to stable states in which the entire population uses {1,1} or {2,2}. (In Model 5, stable variation very near {1,1} or {2,2} is possible.) However, {1,1} and {2,2} are diachronically very stable stress patterns, suggesting that at least for this model set, assuming mistransmission in the learner is problematic. Models 2–3, where analogy is implemented as a hard constraint based on Ross' generalization, do not give most desired properties. Models 4–5, where analogy is implemented as prior probabilities over N/V stress patterns, show crucial aspects of the observed dynamics: bifurcations corresponding to the changes observed in the stress data. Model 5 shows change to {1,2} triggered by falling frequency, a pattern observed in the stress data, and an *emergent* property of the model dynamics: this frequency effect is not present in Models 1 or 4, but is present in Model 5, where the learner combines mistransmission (Model 1) with coupling by priors (Model 4).

5 Discussion

We have developed 5 dynamical systems models for a relatively complex diachronic change, found one successful model, and were able to reason about the source of model behavior. Each model describes the diachronic, population-level consequences of assuming a particular learning algorithm for individuals. The algorithms considered

were motivated by different possible sources of change, from linguistics and psychology (§2.2). We discuss novel contributions of this work, and future directions.

The dataset used here shows more complex dynamics, to our knowledge, than in changes previously considered in the computational literature. By using a detailed, longitudinal dataset, we were able to strongly constrain the desired behavior of a computational model, so that the task of model building is not "doomed to success". While all models show *some* patterns observed in the data, only one shows all such properties. We believe detailed datasets are potentially very useful for evaluating and differentiating between proposed computational models of change.

This paper is a first attempt to integrate detailed data with a range of DS models. We have only considered some schematic properties of the dynamics observed in our dataset, and used these to qualitatively compare each model's predictions to the dynamics. Future work should consider the dynamics in more detail, develop more complex models (for example, by relaxing the infinite-population assumption, allowing for stochastic dynamics), and *quantitatively* compare model predictions and observed dynamics.

We were able to reason about how assumptions about individual learning affect population dynamics by analyzing a range of simple, related models. This approach is pursued in more depth in the larger set of models considered in (Sonderegger, 2009). Our use of model comparison contrasts with most recent computational work on change, where a small number (1–2) of very complex models are analyzed, allowing for much more detailed models of language learning and usage than those considered here (e.g. Choudhury et al., 2006; Minett & Wang, 2008; Baxter et al., 2009; Landsbergen, 2009). An advantage of our approach is an enhanced ability to evaluate a range of proposed causes for a particular case of language change.

By using simple models, we were able to consider a range of learning algorithms corresponding to different explanations for the observed diachronic dynamics. What makes this a useful exercise is the fundamentally non-trivial map, illustrated by Models 1–5, between individual learning and population-level dynamics. Although the type of individual learning assumed in each model was chosen with the same patterns of change in mind, and despite the simplicity of the models used, the resulting population-level dynamics differ greatly. This is an important point given that proposed explanations for change (e.g., mistransmission and analogy) operate at the level of individuals, while the phenomena being explained (patterns of change, or particular changes) are aspects of the population-level dynamics.

Acknowledgments

We thank Max Bane, James Kirby, and three anonymous reviewers for helpful comments.

References

J. Arciuli and L. Cupples. 2003. Effects of stress typicality during speeded grammatical classification. *Language and Speech*, 46(4):353–374.

R.H. Baayen, R. Piepenbrock, and L. Gulikers. 1996. *CELEX2 (CD-ROM)*. Linguistic Data Consortium, Philadelphia.

A. Baker. 2008. Computational approaches to the study of language change. *Language and Linguistics Compass*, 2(3):289–307.

G.J. Baxter, R.A. Blythe, W. Croft, and A.J. McKane. 2009. Modeling language change: An evaluation of Trudgill's theory of the emergence of New Zealand English. *Language Variation and Change*, 21(2):257–296.

J. Blevins. 2006. A theoretical synopsis of Evolutionary Phonology. *Theoretical Linguistics*, 32(2):117–166.

M. Choudhury, A. Basu, and S. Sarkar. 2006. Multi-agent simulation of emergence of schwa deletion pattern in Hindi. *Journal of Artificial Societies and Social Simulation*, 9(2).

M. Choudhury, V. Jalan, S. Sarkar, and A. Basu. 2007. Evolution, optimization, and language change: The case of Bengali verb inflections. In *Proceedings of the Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, pages 65–74.

M. Choudhury. 2007. *Computational Models of Real World Phonological Change*. Ph.D. thesis, Indian Institute of Technology Kharagpur.

R. Daland, A.D. Sims, and J. Pierrehumbert. 2007. Much ado about nothing: A social network model of Russian paradigmatic gaps. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 936–943.

B. de Boer and W. Zuidema. 2009. Models of language evolution: Does the math add up? ILLC Preprint Series PP-2009-49, University of Amsterdam.

T.L. Griffiths and M.L. Kalish. 2007. Language evolution by iterated learning with bayesian agents. *Cognitive Science*, 31(3):441–480.

S.G. Guion, J.J. Clark, T. Harada, and R.P. Wayland. 2003. Factors affecting stress placement for English nonwords include syllabic structure, lexical class, and stress patterns of phonologically similar words. *Language and Speech*, 46(4):403–427.

G.H. Hansson. 2008. Diachronic explanations of sound patterns. *Language & Linguistics Compass*, 2:859–893.

M.W. Hirsch, S. Smale, and R.L. Devaney. 2004. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, Amsterdam, 2nd edition.

H.H. Hock. 1991. *Principles of Historical Linguistics*. Mouton de Gruyter, Berlin, 2nd edition.

M.L. Kalish, T.L. Griffiths, and S. Lewandowsky. 2007. Iterated learning: Intergenerational knowledge transmission reveals inductive biases. *Psychonomic Bulletin and Review*, 14(2):288.

M.H. Kelly and J.K. Bock. 1988. Stress in time. *Journal of Experimental Psychology: Human Perception and Performance*, 14(3):389–403.

M.H. Kelly. 1988. Rhythmic alternation and lexical stress differences in English. *Cognition*, 30:107–137.

M.H. Kelly. 1989. Rhythm and language change in English. *Journal of Memory & Language*, 28:690–710.

S. Kirby, H. Cornish, and K. Smith. 2008. Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686.

S. Klein, M.A. Kuppin, and K.A. Meives. 1969. Monte Carlo simulation of language change in Tikopia & Maori. In *Proceedings of the 1969 Conference on Computational Linguistics*, pages 1–27. ACL.

S. Klein. 1966. Historical change in language using monte carlo techniques. *Mechanical Translation and Computational Linguistics*, 9:67–82.

S. Klein. 1974. Computer simulation of language contact models. In R. Shuy and C-J. Bailey, editors, *Toward Tomorrows Linguistics*, pages 276–290. Georgetown University Press, Washington.

H. Kökeritz. 1953. *Shakespeare's Pronunciation*. Yale University Press, New Haven.

N.L. Komarova, P. Niyogi, and M.A. Nowak. 2001. The evolutionary dynamics of grammar acquisition. *Journal of Theoretical Biology*, 209(1):43–60.

F. Landsbergen. 2009. *Cultural evolutionary modeling of patterns in language change: exercises in evolutionary linguistics*. Ph.D. thesis, Universiteit Leiden.

R. Lass. 1992. Phonology and morphology. In R.M. Hogg, editor, *The Cambridge History of the English Language*, volume 3: 1476–1776, pages 23–156. Cambridge University Press.

P. Levens. 1570. *Manipulus vocabulorum*. Henrie Bynneman, London.

M. MacMahon. 1998. Phonology. In S. Romaine, editor, *The Cambridge History of the English Language*, volume 4: 1476–1776, pages 373–535. Cambridge University Press.

J.W. Minett and W.S.Y. Wang. 2008. Modelling endangered languages: The effects of bilingualism and social structure. *Lingua*, 118(1):19–45.

D. Minkova. 1997. Constraint ranking in Middle English stress-shifting. *English Language and Linguistics*, 1(1):135–175.

W.G. Mitchener. 2005. Simulating language change in the presence of non-idealized syntax. In *Proceedings of the Second Workshop on Psychocomputational Models of Human Language Acquisition*, pages 10–19. ACL.

P. Niyogi and R.C. Berwick. 1995. The logical problem of language change. AI Memo 1516, MIT.

P. Niyogi and R.C. Berwick. 1996. A language learning model for finite parameter spaces. *Cognition*, 61(1-2):161–193.

P. Niyogi. 2006. *The Computational Nature of Language Learning and Evolution*. MIT Press, Cambridge.

J.J. Ohala. 1981. The listener as a source of sound change. In C.S. Masek, R.A. Hendrick, and M.F. Miller, editors, *Papers from the Parasession on Language and Behavior*, pages 178–203. Chicago Linguistic Society, Chicago.

L. Pearl and A. Weinberg. 2007. Input filtering in syntactic acquisition: Answers from language change modeling. *Language Learning and Development*, 3(1):43–72.

B.S. Phillips. 1984. Word frequency and the actuation of sound change. *Language*, 60(2):320–342.

J.R. Ross. 1973. Leftward, ho! In S.R. Anderson and P. Kiparsky, editors, *Festschrift for Morris Halle*, pages 166–173. Holt, Rinehart and Winston, New York.

D. Sherman. 1975. Noun-verb stress alternation: An example of the lexical diffusion of sound change in English. *Linguistics*, 159:43–71.

M. Sonderegger and P. Niyogi. 2010. Variation and change in English noun/verb pair stress: Data, dynamical systems models, and their interaction. Ms. To appear in A.C.L. Yu, editor, *Origins of Sound Patterns: Approaches to Phonologization*. Oxford University Press.

M. Sonderegger. 2009. Dynamical systems models of language variation and change: An application to an English stress shift. Masters paper, Department of Computer Science, University of Chicago.

M. Sonderegger. 2010. Testing for frequency and structural effects in an English stress shift. In *Proceedings of the Berkeley Linguistics Society 36*. To appear.

S. Strogatz. 1994. *Nonlinear Dynamics and Chaos*. Addison-Wesley, Reading, MA.

W.S.Y. Wang, J. Ke, and J.W. Minett. 2005. Computational studies of language evolution. In C. Huang and W. Lenders, editors, *Computational Linguistics and Beyond*, pages 65–108. Institute of Linguistics, Academia Sinica, Taipei.

C. Yang. 2001. Internal and external forces in language change. *Language Variation and Change*, 12(3):231–250.

C. Yang. 2002. *Knowledge and Learning in Natural Language*. Oxford University Press.

Finding Cognate Groups using Phylogenies

David Hall and **Dan Klein**
Computer Science Division
University of California, Berkeley
{dlwh,klein}@cs.berkeley.edu

Abstract

A central problem in historical linguistics is the identification of historically related *cognate* words. We present a generative phylogenetic model for automatically inducing cognate group structure from unaligned word lists. Our model represents the process of transformation and transmission from ancestor word to daughter word, as well as the alignment between the words lists of the observed languages. We also present a novel method for simplifying complex weighted automata created during inference to counteract the otherwise exponential growth of message sizes. On the task of identifying cognates in a dataset of Romance words, our model significantly outperforms a baseline approach, increasing accuracy by as much as 80%. Finally, we demonstrate that our automatically induced groups can be used to successfully reconstruct ancestral words.

1 Introduction

A crowning achievement of historical linguistics is the comparative method (Ohala, 1993), wherein linguists use word similarity to elucidate the hidden phonological and morphological processes which govern historical descent. The comparative method requires reasoning about three important hidden variables: the overall *phylogenetic guide tree* among languages, the *evolutionary parameters* of the ambient changes at each branch, and the *cognate group structure* that specifies which words share common ancestors.

All three of these variables interact and inform each other, and so historical linguists often consider them jointly. However, linguists are currently required to make qualitative judgments regarding the relative likelihood of certain sound changes, cognate groups, and so on. Several recent statistical methods have been introduced to provide increased quantitative backing to the comparative method (Oakes, 2000; Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2009); others have modeled the spread of language changes and speciation (Ringe et al., 2002; Daumé III and Campbell, 2007; Daumé III, 2009; Nerbonne, 2010). These automated methods, while providing robustness and scale in the induction of ancestral word forms and evolutionary parameters, assume that cognate groups are already known. In this work, we address this limitation, presenting a model in which cognate groups can be discovered automatically.

Finding cognate groups is not an easy task, because underlying morphological and phonological changes can obscure relationships between words, especially for distant cognates, where simple string overlap is an inadequate measure of similarity. Indeed, a standard string similarity metric like Levenshtein distance can lead to false positives. Consider the often cited example of Greek /maːti/ and Malay /mata/, both meaning "eye" (Bloomfield, 1938). If we were to rely on Levenshtein distance, these words would seem to be a highly attractive match as cognates: they are nearly identical, essentially differing in only a single character. However, no linguist would posit that these two words are related. To correctly learn that they are not related, linguists typically rely on two kinds of evidence. First, because sound change is largely regular, we would need to commonly see /i/ in Greek wherever we see /a/ in Malay (Ross, 1950). Second, we should look at languages closely related to Greek and Malay, to see if similar patterns hold there, too.

Some authors have attempted to automatically detect cognate words (Mann and Yarowsky, 2001; Lowe and Mazaudon, 1994; Oakes, 2000; Kondrak, 2001; Mulloni, 2007), but these methods

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1030–1039,
Uppsala, Sweden, 11-16 July 2010.

typically work on language pairs rather than on larger language families. To fully automate the comparative method, it is necessary to consider multiple languages, and to do so in a model which couples cognate detection with similarity learning.

In this paper, we present a new generative model for the automatic induction of cognate groups given only (1) a known family tree of languages and (2) word lists from those languages. A prior on word survival generates a number of cognate groups and decides which groups are attested in each modern language. An evolutionary model captures how each word is generated from its parent word. Finally, an alignment model maps the flat word lists to cognate groups. Inference requires a combination of message-passing in the evolutionary model and iterative bipartite graph matching in the alignment model.

In the message-passing phase, our model encodes distributions over strings as weighted finite state automata (Mohri, 2009). Weighted automata have been successfully applied to speech processing (Mohri et al., 1996) and more recently to morphology (Dreyer and Eisner, 2009). Here, we present a new method for automatically compressing our message automata in a way that can take into account prior information about the expected outcome of inference.

In this paper, we focus on a transcribed word list of 583 cognate sets from three Romance languages (Portuguese, Italian and Spanish), as well as their common ancestor Latin (Bouchard-Côté et al., 2007). We consider both the case where we know that all cognate groups have a surface form in all languages, and where we do not know that. On the former, easier task we achieve identification accuracies of 90.6%. On the latter task, we achieve F1 scores of 73.6%. Both substantially beat baseline performance.

2 Model

In this section, we describe a new generative model for vocabulary lists in multiple related languages given the phylogenetic relationship between the languages (their family tree). The generative process factors into three subprocesses: survival, evolution, and alignment, as shown in Figure 1(a). Survival dictates, for each cognate group, which languages have words in that group. Evolution describes the process by which daughter words are transformed from their parent word. Finally, alignment describes the "scrambling" of the word lists into a flat order that hides their lineage. We present each subprocess in detail in the following subsections.

2.1 Survival

First, we choose a number G of ancestral cognate groups from a geometric distribution. For each cognate group g, our generative process walks down the tree. At each branch, the word may either survive or die. This process is modeled in a "death tree" with a Bernoulli random variable $S_{\ell g}$ for each language ℓ and cognate group g specifying whether or not the word died before reaching that language. Death at any node in the tree causes all of that node's descendants to also be dead. This process captures the intuition that cognate words are more likely to be found clustered in sibling languages than scattered across unrelated languages.

2.2 Evolution

Once we know which languages will have an attested word and which will not, we generate the actual word forms. The evolution component of the model generates words according to a branch-specific transformation from a node's immediate ancestor. Figure 1(a) graphically describes our generative model for three Romance languages: Italian, Portuguese, and Spanish.[1] In each cognate group, each word W_ℓ is generated from its parent according to a conditional distribution with parameter φ_ℓ, which is specific to that edge in the tree, but shared between all cognate groups.

In this paper, each φ_ℓ takes the form of a parameterized edit distance similar to the standard Levenshtein distance. Richer models – such as the ones in Bouchard-Côté et al. (2007) – could instead be used, although with an increased inferential cost. The edit transducers are represented schematically in Figure 1(b). Characters x and y are arbitrary phonemes, and $\sigma(x, y)$ represents the cost of substituting x with y. ε represents the empty phoneme and is used as shorthand for insertion and deletion, which have parameters η and δ, respectively.

As an example, see the illustration in Figure 1(c). Here, the Italian word /fwɔko/ ("fire") is generated from its parent form /fokus/ ("hearth")

[1] Though we have data for Latin, we treat it as unobserved to represent the more common case where the ancestral language is unattested; we also evaluate our system using the Latin data.

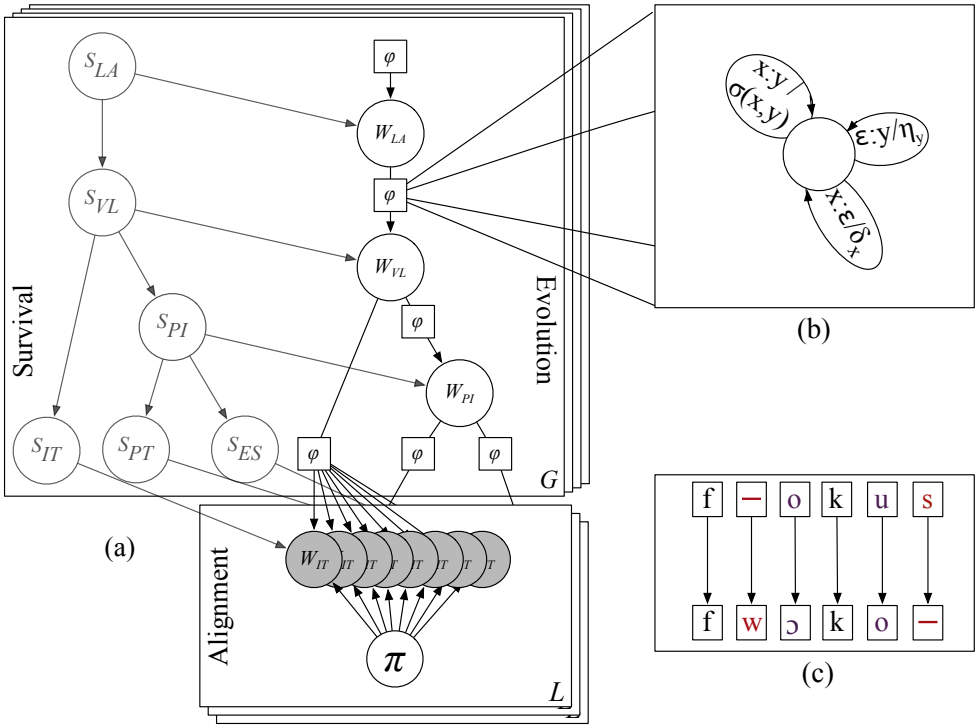


Figure 1: (a) The process by which cognate words are generated. Here, we show the derivation of Romance language words W_ℓ from their respective Latin ancestor, parameterized by transformations φ_ℓ and survival variables S_ℓ. Languages shown are Latin (LA), Vulgar Latin (VL), Proto-Iberian (PI), Italian (IT), Portuguese (PT), and Spanish (ES). Note that only modern language words are observed (shaded). (b) The class of parameterized edit distances used in this paper. Each pair of phonemes has a weight σ for deletion, and each phoneme has weights η and δ for insertion and deletion respectively. (c) A possible alignment produced by an edit distance between the Latin word *focus* ("hearth") and the Italian word *fuoco* ("fire").

by a series of edits: two matches, two substitutions (/u/→ /o/, and /o/→/ɔ/), one insertion (w) and one deletion (/s/). The probability of each individual edit is determined by φ. Note that the marginal probability of a specific Italian word conditioned on its Vulgar Latin parent is the sum over all possible derivations that generate it.

2.3 Alignment

Finally, at the leaves of the trees are the observed words. (We take non-leaf nodes to be unobserved.) Here, we make the simplifying assumption that in any language there is at most one word per language per cognate group. Because the assignments of words to cognates is unknown, we specify an unknown alignment parameter π_ℓ for each modern language which is an alignment of cognate groups to entries in the word list. In the case that every cognate group has a word in each language, each π_ℓ is a permutation. In the more general case that some cognate groups do not have words from all languages, this mapping is injective from words to cognate groups. From a generative perspective, π_ℓ generates observed positions of the words in some vocabulary list.

In this paper, our task is primarily to learn the alignment variables π_ℓ. All other hidden variables are auxiliary and are to be marginalized to the greatest extent possible.

3 Inference of Cognate Assignments

In this section, we discuss the inference method for determining cognate assignments under fixed parameters φ. We are given a set of languages and a list of words in each language, and our objective is to determine which words are cognate with each other. Because the parameters π_ℓ are either permutations or injections, the inference task is reduced to finding an alignment π of the respective word lists to maximize the log probability of the observed words.

$$\pi^* = \arg\max_{\pi} \sum_{g} \log p(w_{(\ell,\pi_\ell(g))} | \varphi, \pi, \mathbf{w}_{-\ell})$$

$w_{(\ell,\pi_\ell(g))}$ is the word in language ℓ that π_ℓ has assigned to cognate group g. Maximizing this quantity directly is intractable, and so instead we use a coordinate ascent algorithm to iteratively

maximize the alignment corresponding to a single language ℓ while holding the others fixed:

$$\pi_\ell^* = \arg\max_{\pi_\ell} \sum_g \log p(w_{(\ell,\pi_\ell(g))}|\varphi, \pi_{-\ell}, \pi_\ell, \mathbf{w}_{-\ell})$$

Each iteration is then actually an instance of bipartite graph matching, with the words in one language one set of nodes, and the current cognate groups in the other languages the other set of nodes. The edge affinities $\mathbf{aff}$ between these nodes are the conditional probabilities of each word w_ℓ belonging to each cognate group g:

$$\mathbf{aff}(w_\ell, g) = p(w_\ell|\mathbf{w}_{-\ell,\pi_{-\ell}(g)}, \varphi, \pi_{-\ell})$$

To compute these affinities, we perform inference in each tree to calculate the marginal distribution of the words from the language ℓ. For the marginals, we use an analog of the forward/backward algorithm. In the upward pass, we send messages from the leaves of the tree toward the root. For observed leaf nodes W_d, we have:

$$\mu_{d\to a}(w_a) = p(W_d = w_d|w_a, \varphi_d)$$

and for interior nodes W_i:

$$\mu_{i\to a}(w_a) = \sum_{w_i} p(w_i|w_a, \varphi_i) \prod_{d\in \text{child}(w_i)} \mu_{d\to i}(w_i) \quad (1)$$

In the downward pass (toward the language ℓ), we sum over ancestral words W_a:

$$\mu_{a\to d}(w_d) = \sum_{w_a} p(w_d|w_a, \varphi_d)\mu_{a'\to a}(w_a) \prod_{\substack{d'\in \text{child}(w_a)\\ d'\neq d}} \mu_{d'\to a}(w_a)$$

where a' is the ancestor of a. Computing these messages gives a posterior marginal distribution $\mu_\ell(w_\ell) = p(w_\ell|\mathbf{w}_{-\ell,\pi_{-\ell}(g)}, \varphi, \pi_{-\ell})$, which is precisely the affinity score we need for the bipartite matching. We then use the Hungarian algorithm (Kuhn, 1955) to find the optimal assignment for the bipartite matching problem.

One important final note is initialization. In our early experiments we found that choosing a random starting configuration unsurprisingly led to rather poor local optima. Instead, we started with empty trees, and added in one language per iteration until all languages were added, and then continued iterations on the full tree.

4 Learning

So far we have only addressed searching for Viterbi alignments π under fixed parameters. In practice, it is important to estimate better parametric edit distances φ_ℓ and survival variables S_ℓ. To motivate the need for good transducers, consider the example of English "day" */deɪ/* and Latin "diēs" */diːɛs/*, both with the same meaning. Surprisingly, these words are in no way related, with English "day" probably coming from a verb meaning "to burn" (OED, 1989). However, a naively constructed edit distance, which for example might penalize vowel substitutions lightly, would fail to learn that Latin words that are borrowed into English would not undergo the sound change */ɪ/→/eɪ/*. Therefore, our model must learn not only which sound changes are plausible (e.g. vowels turning into other vowels is more common than vowels turning into consonants), but which changes are appropriate for a given language.[2]

At a high level, our learning algorithm is much like Expectation Maximization with hard assignments: after we update the alignment variables π and thus form new potential cognate sets, we re-estimate our model's parameters to maximize the likelihood of those assignments.[3] The parameters can be learned through standard maximum likelihood estimation, which we detail in this section.

Because we enforce that a word in language d must be dead if its parent word in language a is dead, we just need to learn the conditional probabilities $p(S_d = \text{dead}|S_a = \text{alive})$. Given fixed assignments π, the maximum likelihood estimate can be found by counting the number of "deaths" that occurred between a child and a live parent, applying smoothing – we found adding 0.5 to be reasonable – and dividing by the total number of live parents.

For the transducers φ, we learn parameterized edit distances that model the probabilities of different sound changes. For each φ_ℓ we fit a non-uniform substitution, insertion, and deletion matrix $\sigma(x, y)$. These edit distances define a condi-

[2] We note two further difficulties: our model does not handle "borrowings," which would be necessary to capture a significant portion of English vocabulary; nor can it seamlessly handle words that are inherited later in the evolution of language than others. For instance, French borrowed words from its parent language Latin during the Renaissance and the Enlightenment that have not undergone the same changes as words that evolved "naturally" from Latin. See Bloomfield (1938). Handling these cases is a direction for future research.

[3] Strictly, we can cast this problem in a variational framework similar to mean field where we iteratively maximize parameters to minimize a KL-divergence. We omit details for clarity.

tional exponential family distribution when conditioned on an ancestral word. That is, for any fixed w_a:

$$\sum_{w_d} p(w_d|w_a,\sigma) = \sum_{w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \text{score}(z;\sigma)$$
$$= \sum_{w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \prod_{(x,y)\in z} \sigma(x,y) = 1$$

where $\text{align}(w_a, w_d)$ is the set of possible alignments between the phonemes in words w_a and w_d.

We are seeking the maximum likelihood estimate of each φ, given fixed alignments π:

$$\hat{\varphi}_\ell = \arg\max_{\varphi_\ell} p(\mathbf{w}|\varphi,\pi)$$

To find this maximizer for any given π_ℓ, we need to find a marginal distribution over the edges connecting any two languages a and d. With this distribution, we calculate the expected "alignment unigrams." That is, for each pair of phonemes x and y (or empty phoneme ε), we need to find the quantity:

$$E_{p(w_a,w_d)}[\#(x,y;z)] =$$
$$\sum_{w_a,w_d} \sum_{\substack{z\in \\ \text{align}(w_a,w_d)}} \#(x,y;z)p(z|w_a,w_d)p(w_a,w_d)$$

where we denote $\#(x,y;z)$ to be the number of times the pair of phonemes (x,y) are aligned in alignment z. The exact method for computing these counts is to use an expectation semiring (Eisner, 2001).

Given the expected counts, we now need to normalize them to ensure that the transducer represents a conditional probability distribution (Eisner, 2002; Oncina and Sebban, 2006). We have that, for each phoneme x in the ancestor language:

$$\eta_y = \frac{E[\#(\varepsilon,y;z)]}{E[\#(\cdot,\cdot;z)]}$$
$$\sigma(x,y) = (1-\sum_{y'}\eta_{y'})\frac{E[\#(x,y;z)]}{E[\#(x,\cdot;z)]}$$
$$\delta_x = (1-\sum_{y'}\eta_{y'})\frac{E[\#(x,\varepsilon;z)]}{E[\#(x,\cdot;z)]}$$

Here, we have $\#(\cdot,\cdot;z) = \sum_{x,y}\#(x,y;z)$ and $\#(x,\cdot;z) = \sum_y \#(x,y;z)$. The $(1-\sum_{y'}\eta_{y'})$ term ensure that for any ancestral phoneme x, $\sum_y \eta_y + \sum_y \sigma(x,y) + \delta_x = 1$. These equations ensure that the three transition types (insertion, substitution/match, deletion) are normalized for each ancestral phoneme.

5 Transducers and Automata

In our model, it is not just the edit distances that are finite state machines. Indeed, the words themselves are string-valued random variables that have, in principle, an infinite domain. To represent distributions and messages over these variables, we chose weighted finite state automata, which can compactly represent functions over strings. Unfortunately, while initially compact, these automata become unwieldy during inference, and so approximations must be used (Dreyer and Eisner, 2009). In this section, we summarize the standard algorithms and representations used for weighted finite state transducers. For more detailed treatment of the general transducer operations, we direct readers to Mohri (2009).

A weighted automaton (resp. transducer) encodes a function over strings (resp. pairs of strings) as weighted paths through a directed graph. Each edge in the graph has a real-valued weight[4] and a label, which is a single phoneme in some alphabet Σ or the empty phoneme ε (resp. pair of labels in some alphabet $\Sigma\times\Delta$). The weight of a string is then the sum of all paths through the graph that accept that string.

For our purposes, we are concerned with three fundamental operations on weighted transducers. The first is computing the sum of all paths through a transducer, which corresponds to computing the partition function of a distribution over strings. This operation can be performed in worst-case cubic time (using a generalization of the Floyd-Warshall algorithm). For acyclic or feed-forward transducers, this time can be improved dramatically by using a generalization of Djisktra's algorithm or other related algorithms (Mohri, 2009).

The second operation is the composition of two transducers. Intuitively, composition creates a new transducer that takes the output from the first transducer, processes it through the second transducer, and then returns the output of the second transducer. That is, consider two transducers T_1 and T_2. T_1 has input alphabet Σ and output alphabet Δ, while T_2 has input alphabet Δ and output alphabet Ω. The composition $T_1 \circ T_2$ returns a new transducer over Σ and Ω such that $(T_1 \circ T_2)(x,y) = \sum_u T_1(x,u)\cdot T_2(u,y)$. In this paper, we use composition for marginalization and factor products. Given a factor $f_1(x,u;T_1)$ and an-

[4]The weights can be anything that form a semiring, but for the sake of exposition we specialize to real-valued weights.

other factor $f_2(u, y; T_2)$, composition corresponds to the operation $\psi(x, y) = \sum_u f_1(x, u) f_2(u, y)$. For two messages $\mu_1(w)$ and $\mu_2(w)$, the same algorithm can be used to find the product $\mu(w) = \mu_1(w)\mu_2(w)$.

The third operation is transducer minimization. Transducer composition produces $O(nm)$ states, where n and m are the number of states in each transducer. Repeated compositions compound the problem: iterated composition of k transducers produces $O(n^k)$ states. Minimization alleviates this problem by collapsing indistinguishable states into a single state. Unfortunately, minimization does not always collapse enough states. In the next section we discuss approaches to "lossy" minimization that produce automata that are not exactly the same but are much smaller.

6 Message Approximation

Recall that in inference, when summing out interior nodes w_i we calculated the product over incoming messages $\mu_{d \to i}(w_i)$ (Equation 1), and that these products are calculated using transducer composition. Unfortunately, the maximal number of states in a message is exponential in the number of words in the cognate group. Minimization can only help so much: in order for two states to be collapsed, the distribution over transitions from those states must be indistinguishable. In practice, for the automata generated in our model, minimization removes at most half the states, which is not sufficient to counteract the exponential growth. Thus, we need to find a way to approximate a message $\mu(w)$ using a simpler automata $\tilde{\mu}(w; \theta)$ taken from a restricted class parameterized by θ.

In the context of transducers, previous authors have focused on a combination of n-best lists and unigram back-off models (Dreyer and Eisner, 2009), a schematic diagram of which is in Figure 2(d). For their problem, n-best lists are sensible: their nodes' local potentials already focus messages on a small number of hypotheses. In our setting, however, n-best lists are problematic; early experiments showed that a 10,000-best list for a typical message only accounts for 50% of message log perplexity. That is, the posterior marginals in our model are (at least initially) fairly flat.

An alternative approach might be to simply treat messages as unnormalized probability distributions, and to minimize the KL divergence between some approximating message $\tilde{\mu}(w)$ and the true message $\mu(w)$. However, messages are not always probability distributions and – because the number of possible strings is in principle infinite – they need not sum to a finite number.[5] Instead, we propose to minimize the KL divergence between the "expected" marginal distribution and the approximated "expected" marginal distribution:

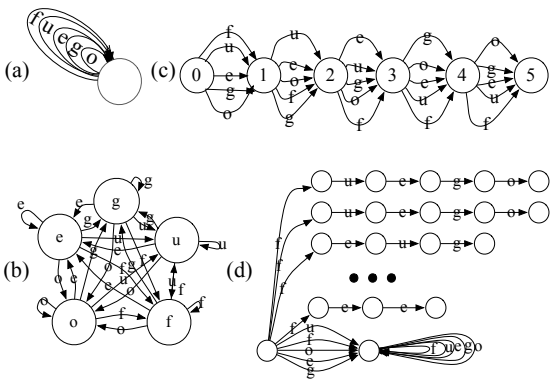


Figure 2: Various topologies for approximating topologies: (a) a unigram model, (b) a bigram model, (c) the anchored unigram model, and (d) the n-best plus backoff model used in Dreyer and Eisner (2009). In (c) and (d), the relative height of arcs is meant to convey approximate probabilities.

$$\begin{aligned}
\hat{\theta} &= \arg\min_{\theta} D_{KL}(\tau(w)\mu(w) || \tau(w)\tilde{\mu}(w; \theta)) \\
&= \arg\min_{\theta} \sum_w \tau(w)\mu(w) \log \frac{\tau(w)\mu(w)}{\tau(w)\tilde{\mu}(w; \theta)} \\
&= \arg\min_{\theta} \sum_w \tau(w)\mu(w) \log \frac{\mu(w)}{\tilde{\mu}(w; \theta)}
\end{aligned} \tag{2}$$

where τ is a term acting as a surrogate for the posterior distribution over w without the information from μ. That is, we seek to approximate μ not on its own, but as it functions in an environment representing its final context. For example, if $\mu(w)$ is a backward message, τ could be a stand-in for a forward probability.[6]

In this paper, $\mu(w)$ is a complex automaton with potentially many states, $\tilde{\mu}(w; \theta)$ is a simple parametric automaton with forms that we discuss below, and $\tau(w)$ is an arbitrary (but hopefully fairly simple) automaton. The actual method we use is

[5] As an extreme example, suppose we have observed that $W_d = w_d$ and that $p(W_d = w_d | w_a) = 1$ for all ancestral words w_a. Then, clearly $\sum_{w_d} \mu(w_d) = \sum_{w_d} \sum p(W_d = w_d | w_a) = \infty$ whenever there are an infinite number of possible ancestral strings w_a.

[6] This approach is reminiscent of Expectation Propagation (Minka, 2001).

as follows. Given a deterministic prior automaton τ, and a deterministic automaton topology $\tilde{\mu}^*$, we create the composed unweighted automaton $\tau \circ \tilde{\mu}^*$, and calculate arc transitions weights to minimize the KL divergence between that composed transducer and $\tau \circ \mu$. The procedure for calculating these statistics is described in Li and Eisner (2009), which amounts to using an expectation semiring (Eisner, 2001) to compute expected transitions in $\tau \circ \tilde{\mu}^*$ under the probability distribution $\tau \circ \mu$.

From there, we need to create the automaton $\tau^{-1} \circ \tau \circ \tilde{\mu}$. That is, we need to divide out the influence of $\tau(w)$. Since we know the topology and arc weights for τ ahead of time, this is often as simple as dividing arc weights in $\tau \circ \tilde{\mu}$ by the corresponding arc weight in $\tau(w)$. For example, if τ encodes a geometric distribution over word lengths and a uniform distribution over phonemes (that is, $\tau(w) \propto p^{|w|}$), then computing $\tilde{\mu}$ is as simple as dividing each arc in $\tau \circ \tilde{\mu}$ by p.[7]

There are a number of choices for τ. One is a hard maximum on the length of words. Another is to choose $\tau(w)$ to be a unigram language model over the language in question with a geometric probability over lengths. In our experiments, we find that $\tau(w)$ can be a geometric distribution over lengths with a uniform distribution over phonemes and still give reasonable results. This distribution captures the importance of shorter strings while still maintaining a relatively weak prior.

What remains is the selection of the topologies for the approximating message $\tilde{\mu}$. We consider three possible approximations, illustrated in Figure 2. The first is a plain unigram model, the second is a bigram model, and the third is an anchored unigram topology: a position-specific unigram model for each position up to some maximum length.

The first we consider is a standard unigram model, which is illustrated in Figure 2(a). It has $|\Sigma| + 2$ parameters: one weight σ_a for each phoneme $a \in \Sigma$, a starting weight λ, and a stopping probability ρ. $\tilde{\mu}$ then has the form:

$$\tilde{\mu}(w) = \lambda\rho \prod_{i \leq |w|} \sigma_{w_i}$$

Estimating this model involves only computing the expected count of each phoneme, along with the expected length of a word, $E[|w|]$. We then normalize the counts according to the maximum likelihood estimate, with arc weights set as:

$$\sigma_a \propto E[\#(a)]$$

Recall that these expectations can be computed using an expectation semiring.

Finally, λ can be computed by ensuring that the approximate and exact expected marginals have the same partition function. That is, with the other parameters fixed, solve:

$$\sum_w \tau(w)\tilde{\mu}(w) = \sum_w \tau(w)\mu(w)$$

which amounts to rescaling $\tilde{\mu}$ by some constant.

The second topology we consider is the bigram topology, illustrated in Figure 2(b). It is similar to the unigram topology except that, instead of a single state, we have a state for each phoneme in Σ, along with a special start state. Each state a has transitions with weights $\sigma_{b|a} = p(b|a) \propto E[\#(b|a)]$. Normalization is similar to the unigram case, except that we normalize the transitions from each state.

The final topology we consider is the positional unigram model in Figure 2(c). This topology takes positional information into account. Namely, for each position (up to some maximum position), we have a unigram model over phonemes emitted at that position, along with the probability of stopping at that position (i.e. a "sausage lattice"). Estimating the parameters of this model is similar, except that the expected counts for the phonemes in the alphabet are conditioned on their position in the string. With the expected counts for each position, we normalize each state's final and outgoing weights. In our experiments, we set the maximum length to seven more than the length of the longest observed string.

7 Experiments

We conduct three experiments. The first is a "complete data" experiment, in which we reconstitute the cognate groups from the Romance data set, where all cognate groups have words in all three languages. This task highlights the evolution and alignment models. The second is a much harder "partial data" experiment, in which we randomly prune 20% of the branches from the dataset according to the survival process described in Section 2.1. Here, only a fraction of words appear

[7] Also, we must be sure to divide each final weight in the transducer by $(1 - |\Sigma|p)$, which is the stopping probability for a geometric transducer.

in any cognate group, so this task crucially involves the survival model. The ultimate purpose of the induced cognate groups is to feed richer evolutionary models, such as full reconstruction models. Therefore, we also consider a proto-word reconstruction experiment. For this experiment, using the system of Bouchard-Côté et al. (2009), we compare the reconstructions produced from our automatic groups to those produced from gold cognate groups.

7.1 Baseline

As a novel but heuristic baseline for cognate group detection, we use an iterative bipartite matching algorithm where instead of conditional likelihoods for affinities we use Dice's coefficient, defined for sets X and Y as:

$$\text{Dice}(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \quad (3)$$

Dice's coefficients are commonly used in bilingual detection of cognates (Kondrak, 2001; Kondrak et al., 2003). We follow prior work and use sets of bigrams within words. In our case, during bipartite matching the set X is the set of bigrams in the language being re-permuted, and Y is the union of bigrams in the other languages.

7.2 Experiment 1: Complete Data

In this experiment, we know precisely how many cognate groups there are and that every cognate group has a word in each language. While this scenario does not include all of the features of the real-world task, it represents a good test case of how well these models can perform without the non-parametric task of deciding how many clusters to use.

We scrambled the 583 cognate groups in the Romance dataset and ran each method to convergence. Besides the heuristic baseline, we tried our model-based approach using Unigrams, Bigrams and Anchored Unigrams, with and without learning the parametric edit distances. When we did not use learning, we set the parameters of the edit distance to (0, -3, -4) for matches, substitutions, and deletions/insertions, respectively. With learning enabled, transducers were initialized with those parameters.

For evaluation, we report two metrics. The first is pairwise accuracy for each pair of languages, averaged across pairs of words. The other is accuracy measured in terms of the number of correctly, completely reconstructed cognate groups.

		Pairwise Acc.	Exact Match
Heuristic			
Baseline		48.1	35.4
Model			
Transducers	Messages		
Levenshtein	Unigrams	37.2	26.2
Levenshtein	Bigrams	43.0	26.5
Levenshtein	Anch. Unigrams	68.6	56.8
Learned	Unigrams	0.1	0.0
Learned	Bigrams	38.7	11.3
Learned	Anch. Unigrams	**90.3**	**86.6**

Table 1: Accuracies for reconstructing cognate groups. Levenshtein refers to fixed parameter edit distance transducer. Learned refers to automatically learned edit distances. Pairwise Accuracy means averaged on each word pair; Exact Match refers to percentage of completely and accurately reconstructed groups. For a description of the baseline, see Section 7.1.

		Prec.	Recall	F1
Heuristic				
Baseline		49.0	43.5	46.1
Model				
Transducers	Messages			
Levenshtein	Anch. Unigrams	**86.5**	36.1	50.9
Learned	Anch. Unigrams	66.9	**82.0**	**73.6**

Table 2: Accuracies for reconstructing incomplete groups. Scores reported are precision, recall, and F1, averaged over all word pairs.

Table 1 shows the results under various configurations. As can be seen, the kind of approximation used matters immensely. In this application, positional information is important, more so than the context of the previous phoneme. Both Unigrams and Bigrams significantly under-perform the baseline, while Anchored Unigrams easily outperforms it both with and without learning.

An initially surprising result is that learning actually harms performance under the unanchored approximations. The explanation is that these topologies are not sensitive enough to context, and that the learning procedure ends up flattening the distributions. In the case of unigrams – which have the least context – learning degrades performance to chance. However, in the case of positional unigrams, learning reduces the error rate by more than two-thirds.

7.3 Experiment 2: Incomplete Data

As a more realistic scenario, we consider the case where we do not know that all cognate groups have words in all languages. To test our model, we ran-

domly pruned 20% of the branches according the survival process of our model.[8]

Because only Anchored Unigrams performed well in Experiment 1, we consider only it and the Dice's coefficient baseline. The baseline needs to be augmented to support the fact that some words may not appear in all cognate groups. To do this, we thresholded the bipartite matching process so that if the coefficient fell below some value, we started a new group for that word. We experimented on 10 values in the range (0,1) for the baseline's threshold and report on the one (0.2) that gives the best pairwise F1.

The results are in Table 2. Here again, we see that the positional unigrams perform much better than the baseline system. The learned transducers seem to sacrifice precision for the sake of increased recall. This makes sense because the default edit distance parameter settings strongly favor exact matches, while the learned transducers learn more realistic substitution and deletion matrices, at the expense of making more mistakes.

For example, the learned transducers enable our model to correctly infer that Portuguese /difemdu/, Spanish /defiendo/, and Italian /difɛndo/ are all derived from Latin /deːfendoː/ "defend." Using the simple Levenshtein transducers, on the other hand, our model keeps all three separated, because the transducers cannot know – among other things – that Portuguese /i/, Spanish /e/, and Italian /i/ are commonly substituted for one another. Unfortunately, because the transducers used cannot learn contextual rules, certain transformations can be over-applied. For instance, Spanish /nombrar/ "name" is grouped together with Portuguese /numirar/ "number" and Italian /numerare/ "number," largely because the rule Portuguese /u/ → Spanish /o/ is applied outside of its normal context. This sound change occurs primarily with final vowels, and does not usually occur word medially. Thus, more sophisticated transducers could learn better sound laws, which could translate into improved accuracy.

7.4 Experiment 3: Reconstructions

As a final trial, we wanted to see how each automatically found cognate group faired as compared to the "true groups" for actual reconstruction of proto-words. Our model is not optimized for faithful reconstruction, and so we used the Ancestry Resampling system of Bouchard-Côté et al. (2009). To evaluate, we matched each Latin word with the best possible cognate group for that word. The process for the matching was as follows. If two or three of the words in an constructed cognate group agreed, we assigned the Latin word associated with the true group to it. With the remainder, we executed a bipartite matching based on bigram overlap.

For evaluation, we examined the Levenshtein distance between the reconstructed word and the chosen Latin word. As a kind of "skyline," we compare to the edit distances reported in Bouchard-Côté et al. (2009), which was based on complete knowledge of the cognate groups. On this task, our reconstructed cognate groups had an average edit distance of 3.8 from the assigned Latin word. This compares favorably to the edit distances reported in Bouchard-Côté et al. (2009), who using oracle cognate assignments achieved an average Levenshtein distance of 3.0.[9]

8 Conclusion

We presented a new generative model of word lists that automatically finds cognate groups from scrambled vocabulary lists. This model jointly models the origin, propagation, and evolution of cognate groups from a common root word. We also introduced a novel technique for approximating automata. Using these approximations, our model can reduce the error rate by 80% over a baseline approach. Finally, we demonstrate that these automatically generated cognate groups can be used to automatically reconstruct proto-words faithfully, with a small increase in error.

Acknowledgments

Thanks to Alexandre Bouchard-Côté for the many insights. This project is funded in part by the NSF under grant 0915265 and an NSF graduate fellowship to the first author.

References

Leonard Bloomfield. 1938. *Language*. Holt, New York.

[8] This dataset will be made available at http://nlp.cs.berkeley.edu/Main.html#Historical

[9] Morphological noise and transcription errors contribute to the absolute error rate for this data set.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *EMNLP*.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved reconstruction of protolanguage word forms. In *NAACL*, pages 65–73.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Conference of the Association for Computational Linguistics (ACL)*.

Hal Daumé III. 2009. Non-parametric Bayesian model areal linguistics. In *NAACL*.

Markus Dreyer and Jason Eisner. 2009. Graphical models over multiple strings. In *EMNLP*, Singapore, August.

Jason Eisner. 2001. Expectation semirings: Flexible EM for finite-state transducers. In Gertjan van Noord, editor, *FSMNLP*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.

Grzegorz Kondrak, Daniel Marcu, and Keven Knight. 2003. Cognates can improve statistical translation models. In *NAACL*.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *NAACL*.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*.

John B. Lowe and Martine Mazaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistics*, 20(3):381–417.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *NAACL*, pages 1–8. Association for Computational Linguistics.

Thomas P. Minka. 2001. Expectation propagation for approximate bayesian inference. In *UAI*, pages 362–369.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *ECAI-96 Workshop*. John Wiley and Sons.

Mehryar Mohri, 2009. *Handbook of Weighted Automata*, chapter Weighted Automata Algorithms. Springer.

Andrea Mulloni. 2007. Automatic prediction of cognate orthography using support vector machines. In *ACL*, pages 25–30.

John Nerbonne. 2010. Measuring the diffusion of linguistic change. *Philosophical Transactions of the Royal Society B: Biological Sciences*.

Michael P. Oakes. 2000. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Quantitative Linguistics*, 7(3):233–243.

OED. 1989. "day, n.". In *The Oxford English Dictionary online*. Oxford University Press.

John Ohala, 1993. *Historical linguistics: Problems and perspectives*, chapter The phonetics of sound change, pages 237–238. Longman.

Jose Oncina and Marc Sebban. 2006. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recognition*, 39(9).

Don Ringe, Tandy Warnow, and Ann Taylor. 2002. Indo-european and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

Alan S.C. Ross. 1950. Philological probability problems. *Journal of the Royal Statistical Society Series B*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *NAACL*.

An Exact A* Method for Deciphering Letter-Substitution Ciphers

Eric Corlett and **Gerald Penn**
Department of Computer Science
University of Toronto
{ecorlett,gpenn}@cs.toronto.edu

Abstract

Letter-substitution ciphers encode a document from a known or hypothesized language into an unknown writing system or an unknown encoding of a known writing system. It is a problem that can occur in a number of practical applications, such as in the problem of determining the encodings of electronic documents in which the language is known, but the encoding standard is not. It has also been used in relation to OCR applications. In this paper, we introduce an exact method for deciphering messages using a generalization of the Viterbi algorithm. We test this model on a set of ciphers developed from various web sites, and find that our algorithm has the potential to be a viable, practical method for efficiently solving decipherment problems.

1 Introduction

Letter-substitution ciphers encode a document from a known language into an unknown writing system or an unknown encoding of a known writing system. This problem has practical significance in a number of areas, such as in reading electronic documents that may use one of many different standards to encode text. While this is not a problem in languages like English and Chinese, which have a small set of well known standard encodings such as ASCII, Big5 and Unicode, there are other languages such as Hindi in which there is no dominant encoding standard for the writing system. In these languages, we would like to be able to automatically retrieve and display the information in electronic documents which use unknown encodings when we find them. We also want to use these documents for information retrieval and data mining, in which case it is important to be able to read through them automatically, without resorting to a human annotator. The holy grail in this area would be an application to archaeological decipherment, in which the underlying language's identity is only hypothesized, and must be tested. The purpose of this paper, then, is to simplify the problem of reading documents in unknown encodings by presenting a new algorithm to be used in their decipherment. Our algorithm operates by running a search over the n-gram probabilities of possible solutions to the cipher, using a generalization of the Viterbi algorithm that is wrapped in an A* search, which determines at each step which partial solutions to expand. It is guaranteed to converge on the language-model-optimal solution, and does not require restarts or risk falling into local optima. We specifically consider the problem of finding decodings of electronic documents drawn from the internet, and we test our algorithm on ciphers drawn from randomly selected pages of Wikipedia. Our testing indicates that our algorithm will be effective in this domain.

It may seem at first that automatically decoding (as opposed to deciphering) a document is a simple matter, but studies have shown that simple algorithms such as letter frequency counting do not always produce optimal solutions (Bauer, 2007). If the text from which a language model is trained is of a different genre than the plaintext of a cipher, the unigraph letter frequencies may differ substantially from those of the language model, and so frequency counting will be misleading. Because of the perceived simplicity of the problem, however, little work was performed to understand its computational properties until Peleg and Rosenfeld (1979), who developed a method that repeatedly swaps letters in a cipher to find a maximum probability solution. Since then, several different approaches to this problem have been suggested, some of which use word counts in the language to arrive at a solution (Hart, 1994), and some of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1040–1047,
Uppsala, Sweden, 11-16 July 2010.

which treat the problem as an expectation maximization problem (Knight et al., 2006; Knight, 1999). These later algorithms are, however, highly dependent on their initial states, and require a number of restarts in order to find the globally optimal solution. A further contribution was made by (Ravi and Knight, 2008), which, though published earlier, was inspired in part by the method presented here, first discovered in 2007. Unlike the present method, however, Ravi and Knight (2008) treat the decipherment of letter-substitution ciphers as an integer programming problem. Clever though this constraint-based encoding is, their paper does not quantify the massive running times required to decode even very short documents with this sort of approach. Such inefficiency indicates that integer programming may simply be the wrong tool for the job, possibly because language model probabilities computed from empirical data are not smoothly distributed enough over the space in which a cutting-plane method would attempt to compute a linear relaxation of this problem. In any case, an exact method is available with a much more efficient A* search that is linear-time in the length of the cipher (though still horribly exponential in the size of the cipher and plain text alphabets), and has the additional advantage of being massively parallelizable. (Ravi and Knight, 2008) also seem to believe that short cipher texts are somehow inherently more difficult to solve than long cipher texts. This difference in difficulty, while real, is not inherent, but rather an artefact of the character-level n-gram language models that they (and we) use, in which preponderant evidence of differences in short character sequences is necessary for the model to clearly favour one letter-substitution mapping over another. Uniform character models equivocate regardless of the length of the cipher, and sharp character models with many zeroes can quickly converge even on short ciphers of only a few characters. In the present method, the role of the language model can be acutely perceived; both the time complexity of the algorithm and the accuracy of the results depend crucially on this characteristic of the language model. In fact, we must use add-one smoothing to decipher texts of even modest lengths because even one unseen plain-text letter sequence is enough to knock out the correct solution. It is likely that the method of (Ravi and Knight, 2008) is sensitive to this as well, but their experiments were apparently fixed on a single, well-trained model.

Applications of decipherment are also explored by (Nagy et al., 1987), who uses it in the context of optical character recognition (OCR). The problem we consider here is cosmetically related to the "L2P" (letter-to-phoneme) mapping problem of text-to-speech synthesis, which also features a prominent constraint-based approach (van den Bosch and Canisius, 2006), but the constraints in L2P are very different: two different instances of the same written letter may legitimately map to two different phonemes. This is not the case in letter-substitution maps.

2 Terminology

Substitution ciphers are ciphers that are defined by some permutation of a plaintext alphabet. Every character of a plaintext string is consistently mapped to a single character of an output string using this permutation. For example, if we took the string "$hello\_world$" to be the plaintext, then the string "$ifmmp\_xpsme$" would be a cipher that maps e to f, l to m, and so on. It is easy to extend this kind of cipher so that the plaintext alphabet is different from the ciphertext alphabet, but still stands in a one to one correspondence to it. Given a ciphertext C, we say that the set of characters used in C is the ciphertext alphabet Σ_C, and that its size is n_C. Similarly, the entire possible plaintext alphabet is Σ_P, and its size is is n_P. Since n_C is the number of letters actually used in the cipher, rather than the entire alphabet it is sampled from, we may find that $n_C < n_P$ even when the two alphabets are the same. We refer to the length of the cipher string C as c_{len}. In the above example, Σ_P is $\{\_, a, \ldots z\}$ and $n_P = 27$, while $\Sigma_C = \{\_, e, f, i, m, p, s, x\}$, $c_{len} = 11$ and $n_C = 8$.

Given the ciphertext C, we say that a *partial solution* of size k is a map $\sigma = \{p_1 : c_1, \ldots p_k : c_k\}$, where $c_1, \ldots, c_k \in \Sigma_C$ and are distinct, and $p_1, \ldots, p_k \in \Sigma_P$ and are distinct, and where $k \leq n_C$. If for a partial solution σ', we have that $\sigma \subset \sigma'$, then we say that σ' *extends* σ. If the size of σ' is $k+1$ and σ is size k, we say that σ' is an *immediate extension* of σ. A *full solution* is a partial solution of size n_C. In the above example, $\sigma_1 = \{\_ : \_, d : e\}$ would be a partial solution of size 2, and $\sigma_2 = \{\_ : \_, d : e, g : m\}$ would be a partial solution of size 3 that immediately extends σ_1. A partial solution $\sigma_T\{\_ : \_, d : e, e : f, h : i, l : m, o :$

$p, r : s, w : x\}$ would be both a full solution and the correct one. The full solution σ_T extends σ_1 but not σ_2.

Every possible full solution to a cipher C will produce a plaintext string with some associated language model probability, and we will consider the best possible solution to be the one that gives the highest probability. For the sake of concreteness, we will assume here that the language model is a character-level trigram model. This plaintext can be found by treating all of the length c_{len} strings S as being the output of different character mappings from C. A string S that results from such a mapping is *consistent* with a partial solution σ iff, for every $p_i : c_i \in \sigma$, the character positions of C that map to p_i are exactly the character positions with c_i in C.

In our above example, we had $C =$ "$ifmmp\_xpsme$", in which case we had $c_{len} = 11$. So mappings from C to "$hhhhh\_hhhhh$" or "$\_hhhhhhhhhh$" would be consistent with a partial solution of size 0, while "$hhhhh\_hhhhn$" would be consistent with the size 2 partial solution $\sigma = \{\_ : \_, n : e\}$.

3 The Algorithm

In order to efficiently search for the most likely solution for a ciphertext C, we conduct a search of the partial solutions using their trigram probabilities as a heuristic, where the trigram probability of a partial solution σ of length k is the maximum trigram probability over all strings consistent with it, meaning, in particular, that ciphertext letters not in its range can be mapped to any plaintext letter, and do not even need to be consistently mapped to the same plaintext letter in every instance. Given a partial solution σ of length n, we can extend σ by choosing a ciphertext letter c not in the range of σ, and then use our generalization of the Viterbi algorithm to find, for each p not in the domain of σ, a score to rank the choice of p for c, namely the trigram probability of the extension σ_p of σ. If we start with an empty solution and iteratively choose the most likely remaining partial solution in this way, storing the extensions obtained in a priority heap as we go, we will eventually reach a solution of size n_C. Every extension of σ has a probability that is, at best, equal to that of σ, and every partial solution receives, at worst, a score equal to its best extension, because the score is potentially based on an inconsistent mapping that does not qualify as an extension. These two observations taken together mean that one minus the score assigned by our method constitutes a cost function over which this score is an admissible heuristic in the A* sense. Thus the first solution of size n_C will be the best solution of size n_C.

The order by which we add the letters c to partial solutions is the order of the distinct ciphertext letters in right-to-left order of their final occurrence in C. Other orderings for the c, such as most frequent first, are also possible though less elegant.[1]

Algorithm 1 Search Algorithm

Order the letters $c_1 \ldots c_{n_C}$ by rightmost occurrence in C, $r_{n_C} < \ldots < r_1$.
Create a priority queue Q for partial solutions, ordered by highest probability.
Push the empty solution $\sigma_0 = \{\}$ onto the queue.
while Q is not empty **do**
  Pop the best partial solution σ from Q.
  $s = |\sigma|$.
  if $s = n_C$ **then**
    return σ
  else
    For all p not in the range of σ, push the immediate extension σ_p onto Q with the score assigned to table cell $G(r_{s+1}, p, p)$ by GVit$(\sigma, c_{s+1}, r_{s+1})$ if it is non-zero.
  end if
end while
Return "Solution Infeasible".

Our generalization of the Viterbi algorithm, depicted in Figure 1, uses dynamic programming to score every immediate extension of a given partial solution in tandem, by finding, in a manner consistent with the real Viterbi algorithm, the most probable input string given a set of output symbols, which in this case is the cipher C. Unlike the real Viterbi algorithm, we must also observe the constraints of the input partial solution's mapping.

[1] We have experimented with the most frequent first regimen as well, and it performs worse than the one reported here. Our hypothesis is that this is due to the fact that the most frequent character tends to appear in many high-frequency trigrams, and so our priority queue becomes very long because of a lack of low-probability trigrams to knock the scores of partial solutions below the scores of the extensions of their better scoring but same-length peers. A least frequent first regimen has the opposite problem, in which their rare occurrence in the ciphertext provides too few opportunities to potentially reduce the score of a candidate.

A typical decipherment involves multiple runs of this algorithm, each of which scores all of the immediate extensions, both tightening and lowering their scores relative to the score of the input partial solution. A call GVit(σ, c, r) manages this by filling in a table G such that for all $1 \leq i \leq r$, and $l, k \in \Sigma_P$, $G(i, l, k)$ is the maximum probability over every plaintext string S for which:

- len$(S) = i$,
- $S[i] = l$,
- for every p in the domain of σ, every $1 \leq j \leq i$, if $C[j] = \sigma(p)$ then $S[j] = p$, and
- for every position $1 \leq j \leq i$, if $C[j] = c$, then $S[j] = k$.

The real Viterbi algorithm lacks these final two constraints, and would only store a single cell at $G(i, l)$. There, G is called a trellis. Ours is larger, so so we will refer to G as a *greenhouse*.

The table is completed by filling in the columns from $i = 1$ to c_{len} in order. In every column i, we will iterate over the values of l and over the values of k such that $k : c$ and $l :$ are consistent with σ. Because we are using a trigram character model, the cells in the first and second columns must be primed with unigram and bigram probabilities. The remaining probabilities are calculated by searching through the cells from the previous two columns, using the entry at the earlier column to indicate the probability of the best string up to that point, and searching through the trigram probabilities over two additional letters. Backpointers are necessary to reference one of the two language model probabilities. Cells that would produce inconsistencies are left at zero, and these as well as cells that the language model assigns zero to can only produce zero entries in later columns.

In order to decrease the search space, we add the further restriction that the solutions of every three character sequence must be consistent: if the ciphertext indicates that two adjacent letters are the same, then only the plaintext strings that map the same letter to each will be considered. The number of letters that are forced to be consistent is three because consistency is enforced by removing inconsistent strings from consideration during trigram model evaluation.

Because every partial solution is only obtained by extending a solution of size one less, and extensions are only made in a predetermined order of cipher alphabet letters, every partial solution is only considered / extended once.

GVit is highly parallelizable. The $n_P \times n_P$ cells of every column i do not depend on each other — only on the cells of the previous two columns $i-1$ and $i-2$, as well as the language model. In our implementation of the algorithm, we have written the underlying program in C/C++, and we have used the CUDA library developed for NVIDIA graphics cards to in order to implement the parallel sections of the code.

4 Experiment

The above algorithm is designed for application to the transliteration of electronic documents, specifically, the transliteration of websites, and it has been tested with this in mind. In order to gain realistic test data, we have operated on the assumption that Wikipedia is a good approximation of the type of language that will be found in most internet articles. We sampled a sequence of English-language articles from Wikipedia using their random page selector, and these were used to create a set of reference pages. In order to minimize the common material used in each page, only the text enclosed by the paragraph tags of the main body of the pages were used. A rough search over internet articles has shown that a length of 1000 to 11000 characters is a realistic length for many articles, although this can vary according to the genre of the page. Wikipedia, for example, does have entries that are one sentence in length. We have run two groups of tests for our algorithm. In the first set of tests, we chose the mean of the above lengths to be our sample size, and we created and decoded 10 ciphers of this size (i.e., different texts, same size). We made these cipher texts by appending the contents of randomly chosen Wikipedia pages until they contained at least 6000 characters, and then using the first 6000 characters of the resulting files as the plaintexts of the cipher. The text length was rounded up to the nearest word where needed. In the second set of tests, we used a single long ciphertext, and measured the time required for the algorithm to finish a number of prefixes of it (i.e., same text, different sizes). The plaintext for this set of tests was developed in the same way as the first set, and the input ciphertext lengths considered were 1000, 3500, 6000, 8500, 11000, and 13500 characters.

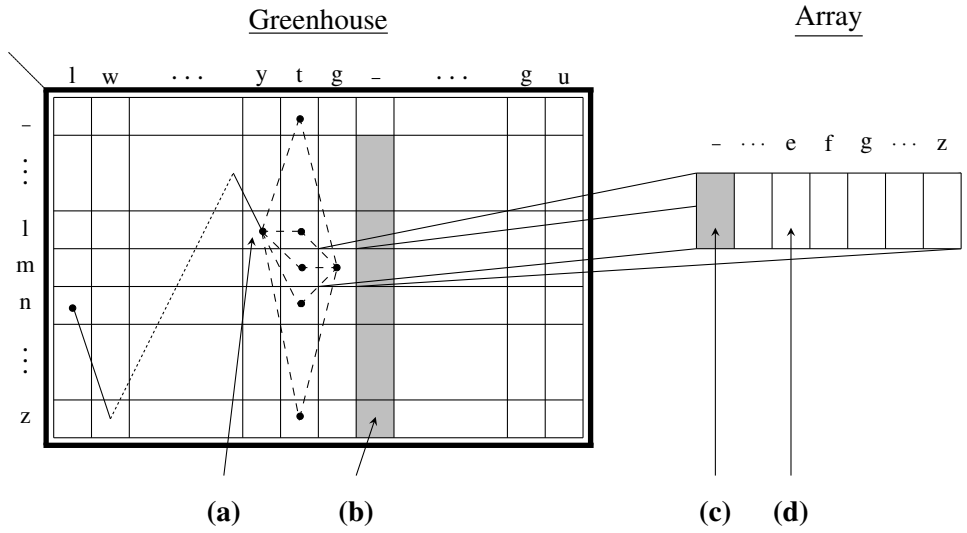


Figure 1: Filling the Greenhouse Table. Each cell in the greenhouse is indexed by a plaintext letter and a character from the cipher. Each cell consists of a smaller array. The cells in the array give the best probabilities of any path passing through the greenhouse cell, given that the index character of the array maps to the character in column c, where c is the next ciphertext character to be fixed in the solution. The probability is set to zero if no path can pass through the cell. This is the case, for example, in (b) and (c), where the knowledge that "_" maps to "_" would tell us that the cells indicated in gray are unreachable. The cell at (d) is filled using the trigram probabilities and the probability of the path at starting at (a).

In all of the data considered, the frequency of spaces was far higher than that of any other character, and so in any real application the character corresponding to the space can likely be guessed without difficulty. The ciphers we have considered have therefore been simplified by allowing the knowledge of which character corresponds to the space. It appears that Ravi and Knight (2008) did this as well. Our algorithm will still work without this assumption, but would take longer. In the event that a trigram or bigram would be found in the plaintext that was not counted in the language model, add one smoothing was used.

Our character-level language model used was developed from the first 1.5 million characters of the Wall Street Journal section of the Penn Treebank corpus. The characters used in the language model were the upper and lower case letters, spaces, and full stops; other characters were skipped when counting the frequencies. Furthermore, the number of sequential spaces allowed was limited to one in order to maximize context and to eliminate any long stretches of white space. As discussed in the previous paragraph, the space character is assumed to be known.

When testing our algorithm, we judged the time complexity of our algorithm by measuring the actual time taken by the algorithm to complete its runs, as well as the number of partial solutions placed onto the queue ("enqueued"), the number popped off the queue ("expanded"), and the number of zero-probability partial solutions not enqueued ("zeros") during these runs. These latter numbers give us insight into the quality of trigram probabilities as a heuristic for the A* search.

We judged the quality of the decoding by measuring the percentage of characters in the cipher alphabet that were correctly guessed, and also the word error rate of the plaintext generated by our solution. The second metric is useful because a low probability character in the ciphertext may be guessed wrong without changing as much of the actual plaintext. Counting the actual number of word errors is meant as an estimate of how useful or readable the plaintext will be. We did not count the accuracy or word error rate for unfinished ciphers.

We would have liked to compare our results with those of Ravi and Knight (2008), but the method presented there was simply not feasible

Algorithm 2 Generalized Viterbi Algorithm

GVit(σ, c, r)
Input: partial solution σ, ciphertext character c, and index r into C.
Output: greenhouse G.
Initialize G to 0.
i = 1
for All (l, k) such that $\sigma \cup \{k : c, l : C_i\}$ is consistent **do**
  $G(i, l, k) = P(l)$.
end for
i = 2
for All (l, k) such that $\sigma \cup \{k : c, l : C_i\}$ is consistent **do**
  for j such that $\sigma \cup \{k : c, l : C_i, j : C_{i-1}\}$ is consistent **do**
    $G(i, l, k) = \max(G(i, l, k), G(0, j, k) \times P(l|j))$
  end for
end for
i = 3
for (l, k) such that $\sigma \cup \{k : c, l : C_i\}$ is consistent **do**
  for j_1, j_2 such that $\sigma \cup \{k : c, j_2 : C[i-2], j_1 : C[i-1], l : C_i\}$ is consistent **do**
    $G(i, l, k) = \max(G(i, l, k), G(i-2, j_2, k) \times P(j_1|j_2) \times P(l|j_2 j_1))$.
  end for
end for
for $i = 4$ to r **do**
  for (l, k) such that $\sigma \cup \{k : c, l : C_i\}$ is consistent **do**
    for j_1, j_2 such that $\sigma \cup \{k : c, j_2 : C[i-2], j_1 : C[i-1], l : C_i\}$ is consistent **do**
      $G(i, l, k) = \max(G(i, l, k), G(i-2, j_2, k) \times P(j_1|j_2 j_{2(back)}) \times P(l|j_2 j_1))$.
    end for
  end for
end for

on texts and (case-sensitive) alphabets of this size with the computing hardware at our disposal.

5 Results

In our first set of tests, we measured the time consumption and accuracy of our algorithm over 10 ciphers taken from random texts that were 6000 characters long. The time values in these tables are given in the format of (H)H:MM:SS. For this set of tests, in the event that a test took more than 12 hours, we terminated it and listed it as unfinished. This cutoff was set in advance of the runs based upon our armchair speculation about how long one might at most be reasonably expected to wait for a web-page to be transliterated (an overnight run). The results from this run appear in Table 1. All running times reported in this section were obtained on a computer running Ubuntu Linux 8.04 with 4 GB of RAM and 8×2.5 GHz CPU cores. Column-level subcomputations in the greenhouse were dispatched to an NVIDIA Quadro FX 1700 GPU card that is attached through a 16-lane PCI Express adapter. The card has 512 MB of cache memory, a 460 MHz core processor and 32 shader processors operating in parallel at 920 MHz each.

In our second set of tests, we measured the time consumption and accuracy of our algorithm over several prefixes of different lengths of a single 13500-character ciphertext. The results of this run are given in Table 2.

The first thing to note in this data is that the accuracy of this algorithm is above 90 % for all of the test data, and 100% on all but the smallest 2 ciphers. We can also observe that even when there are errors (e.g., in the size 1000 cipher), the word error rate is very small. This is a Zipf's Law effect — misclassified characters come from poorly attested character trigrams, which are in turn found only in longer, rarer words. The overall high accuracy is probably due to the large size of the texts relative to the uniticity distance of an English letter-substitution cipher (Bauer, 2007). The results do show, however, that character trigram probabilities are an effective indicator of the most likely solution, even when the language model and test data are from very different genres (here, the Wall Street Journal and Wikipedia, respectively). These results also show that our algorithm is effective as a way of decoding simple ciphers. 80% of our runs finished before the 12 hour cutoff in the first experiment.

Cipher	Time	Enqueued	Expanded	Zeros	Accuracy	Word Error Rate
1	2:03:06	964	964	44157	100%	0%
2	0:13:00	132	132	5197	100%	0%
3	0:05:42	91	91	3080	100%	0%
4	Unfinished	N/A	N/A	N/A	N/A	N/A
5	Unfinished	N/A	N/A	N/A	N/A	N/A
6	5:33:50	2521	2521	114283	100%	0%
7	6:02:41	2626	2626	116392	100%	0%
8	3:19:17	1483	1483	66070	100%	0%
9	9:22:54	4814	4814	215086	100%	0%
10	1:23:21	950	950	42107	100%	0%

Table 1: Time consumption and accuracy on a sample of 10 6000-character texts.

Size	Time	Enqueued	Expanded	Zeros	Accuracy	Word Error Rate
1000	40:06:05	119759	119755	5172631	92.59%	1.89%
3500	0:38:02	615	614	26865	96.30%	0.17%
6000	0:12:34	147	147	5709	100%	0%
8500	8:52:25	1302	1302	60978	100%	0%
11000	1:03:58	210	210	8868	100%	0%
13500	0:54:30	219	219	9277	100%	0%

Table 2: Time consumption and accuracy on prefixes of a single 13500-character ciphertext.

As far as the running time of the algorithm goes, we see a substantial variance: from a few minutes to several hours for most of the longer ciphers, and that there are some that take longer than the threshold we gave in the experiment. Specifically, there is substantial variability in the the running times seen.

Desiring to reduce the variance of the running time, we look at the second set of tests for possible causes. In the second test set, there is a general decrease in both the running time and the number of solutions expanded as the length of the ciphers increases. Running time correlates very well with A* queue size.

Asymptotically, the time required for each sweep of the Viterbi algorithm increases, but this is more than offset by the decrease in the number of required sweeps.

The results, however, do not show that running time monotonically decreases with length. In particular, the length 8500 cipher generates more solutions than the length 3500 or 6000 ones. Recall that the ciphers in this section are all prefixes of the same string. Because the algorithm fixes characters starting from the end of the cipher, these prefixes have very different character orderings, $c_1, \ldots, c_{n_C}$, and thus a very different order of partial solutions. The running time of our algorithm depends very crucially on these initial conditions.

Perhaps most interestingly, we note that the number of enqueued partial solutions is in every case identical or nearly identical to the number of partial solutions expanded. From a theoretical perspective, we must also remember the zero-probability solutions, which should in a sense count when judging the effectiveness of our A* heuristic. Naturally, these are ignored by our implementation because they are so badly scored that they could never be considered. Nevertheless, what these numbers show is that scores based on character-level trigrams, while theoretically admissible, are really not all that clever when it comes to navigating through the search space of all possible letter substitution ciphers, apart from their very keen ability at assigning zeros to a large number of partial solutions. A more complex heuristic that can additionally rank non-zero probability solutions with more prescience would likely make a very great difference to the running time of this method.

6 Conclusions

In the above paper, we have presented an algorithm for solving letter-substitution ciphers, with an eye towards discovering unknown encoding standards in electronic documents on the fly. In a test of our algorithm over ciphers drawn from Wikipedia, we found its accuracy to be 100% on the ciphers that it solved within a threshold of 12 hours, this being 80% of the total attempted. We found that the running time of our algorithm is highly variable depending on the order of characters attempted, and, due to the linear-time theoretical complexity of this method, that running times tend to decrease with larger ciphertexts due to our character-level language model's facility at eliminating highly improbable solutions. There is, however, a great deal of room for improvement in the trigram model's ability to rank partial solutions that are not eliminated outright.

Perhaps the most valuable insight gleaned from this study has been on the role of the language model. This algorithm's asymptotic runtime complexity is actually a function of entropic aspects of the character-level language model that it uses — more uniform models provide less prominent separations between candidate partial solutions, and this leads to badly ordered queues, in which extended partial solutions can never compete with partial solutions that have smaller domains, leading to a blind search. We believe that there is a great deal of promise in characterizing natural language processing algorithms in this way, due to the prevalence of Bayesian methods that use language models as priors.

Our approach makes no explicit attempt to account for noisy ciphers, in which characters are erroneously mapped, nor any attempt to account for more general substitution ciphers in which a single plaintext (resp. ciphertext) letter can map to multiple ciphertext (resp. plaintext) letters, nor for ciphers in which ciphertext units corresponds to larger units of plaintext such syllables or words. Extensions in these directions are all very worthwhile to explore.

References

Friedrich L. Bauer. 2007. *Decrypted Secrets*. Springer-Verlag, Berlin Heidelberg.

George W. Hart. 1994. To Decode Short Cryptograms. *Communications of the ACM*, 37(9): 102–108.

Kevin Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, 25(4):607–615.

Kevin Knight, Anish Nair, Nishit Rathod, Kenji Yamada. Unsupervised Analysis for Decipherment Problems. *Proceedings of the COLING/ACL 2006*, 2006, 499–506.

George Nagy, Sharad Seth, Kent Einspahr. 1987. Decoding Substitution Ciphers by Means of Word Matching with Application to OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):710–715.

Shmuel Peleg and Azriel Rosenfeld. 1979. Breaking Substitution Ciphers Using a Relaxation Algorithm. *Communications of the ACM*, 22(11):589–605.

Sujith Ravi, Kevin Knight. 2008. Attacking Decipherment Problems Optimally with Low-Order N-gram Models *Proceedings of the ACL 2008*, 812–819.

Antal van den Bosch, Sander Canisius. 2006. Improved Morpho-phonological Sequence Processing with Constraint Satisfaction Inference *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL 2006*, 41–49.

A Statistical Model for Lost Language Decipherment

Benjamin Snyder and **Regina Barzilay**
CSAIL
Massachusetts Institute of Technology
{bsnyder,regina}@csail.mit.edu

Kevin Knight
ISI
University of Southern California
knight@isi.edu

Abstract

In this paper we propose a method for the automatic decipherment of lost languages. Given a non-parallel corpus in a known related language, our model produces both alphabetic mappings and translations of words into their corresponding cognates. We employ a non-parametric Bayesian framework to simultaneously capture both low-level character mappings and high-level morphemic correspondences. This formulation enables us to encode some of the linguistic intuitions that have guided human decipherers. When applied to the ancient Semitic language Ugaritic, the model correctly maps 29 of 30 letters to their Hebrew counterparts, and deduces the correct Hebrew cognate for 60% of the Ugaritic words which have cognates in Hebrew.

1 Introduction

Dozens of lost languages have been deciphered by humans in the last two centuries. In each case, the decipherment has been considered a major intellectual breakthrough, often the culmination of decades of scholarly efforts. Computers have played no role in the decipherment any of these languages. In fact, skeptics argue that computers do not possess the "logic and intuition" required to unravel the mysteries of ancient scripts.[1] In this paper, we demonstrate that at least some of this logic and intuition *can* be successfully modeled, allowing computational tools to be used in the decipherment process.

[1] *"Successful archaeological decipherment has turned out to require a synthesis of logic and intuition ... that computers do not (and presumably cannot) possess."* A. Robinson, "Lost Languages: The Enigma of the World's Undeciphered Scripts" (2002)

Our definition of the computational decipherment task closely follows the setup typically faced by human decipherers (Robinson, 2002). Our input consists of texts in a lost language and a corpus of non-parallel data in a known related language. The decipherment itself involves two related subtasks: (i) finding the mapping between alphabets of the known and lost languages, and (ii) translating words in the lost language into corresponding cognates of the known language.

While there is no single formula that human decipherers have employed, manual efforts have focused on several guiding principles. A common starting point is to compare letter and word frequencies between the lost and known languages. In the presence of cognates the correct mapping between the languages will reveal similarities in frequency, both at the character and lexical level. In addition, morphological analysis plays a crucial role here, as highly frequent morpheme correspondences can be particularly revealing. In fact, these three strands of analysis (character frequency, morphology, and lexical frequency) are intertwined throughout the human decipherment process. Partial knowledge of each drives discovery in the others.

We capture these intuitions in a generative Bayesian model. This model assumes that each word in the lost language is composed of morphemes which were generated with latent counterparts in the known language. We model bilingual morpheme pairs as arising through a series of Dirichlet processes. This allows us to assign probabilities based both on character-level correspondences (using a character-edit base distribution) as well as higher-level morpheme correspondences. In addition, our model carries out an implicit morphological analysis of the lost language, utilizing the known morphological structure of the related language. This model structure allows us to capture the interplay between the character-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1048–1057,
Uppsala, Sweden, 11-16 July 2010.

and morpheme-level correspondences that humans have used in the manual decipherment process.

In addition, we introduce a novel technique for imposing structural sparsity constraints on character-level mappings. We assume that an accurate alphabetic mapping between related languages will be sparse in the following way: each letter will map to a very limited subset of letters in the other language. We capture this intuition by adapting the so-called "spike and slab" prior to the Dirichlet-multinomial setting. For each pair of characters in the two languages, we posit an indicator variable which controls the prior likelihood of character substitutions. We define a joint prior over these indicator variables which encourages sparse settings.

We applied our model to a corpus of Ugaritic, an ancient Semitic language discovered in 1928. Ugaritic was manually deciphered in 1932, using knowledge of Hebrew, a related language. We compare our method against the only existing decipherment baseline, an HMM-based character substitution cipher (Knight and Yamada, 1999; Knight et al., 2006). The baseline correctly maps the majority of letters — 22 out of 30 — to their correct Hebrew counterparts, but only correctly translates 29% of all cognates. In comparison, our method yields correct mappings for 29 of 30 letters, and correctly translates 60.4% of all cognates.

2 Related Work

Our work on decipherment has connections to three lines of work in statistical NLP. First, our work relates to research on cognate identification (Lowe and Mazaudon, 1994; Guy, 1994; Kondrak, 2001; Bouchard et al., 2007; Kondrak, 2009). These methods typically rely on information that is unknown in a typical deciphering scenario (while being readily available for living languages). For instance, some methods employ a hand-coded similarity function (Kondrak, 2001), while others assume knowledge of the phonetic mapping or require parallel cognate pairs to learn a similarity function (Bouchard et al., 2007).

A second related line of work is lexicon induction from non-parallel corpora. While this research has similar goals, it typically builds on information or resources unavailable for ancient texts, such as comparable corpora, a seed lexicon, and cognate information (Fung and McKeown, 1997; Rapp, 1999; Koehn and Knight, 2002; Haghighi et al., 2008). Moreover, distributional methods that rely on co-occurrence analysis operate over large corpora, which are typically unavailable for a lost language.

Finally, Knight and Yamada (1999) and Knight et al. (2006) describe a computational HMM-based method for deciphering an unknown script that represents a known spoken language. This method "makes the text speak" by gleaning character-to-sound mappings from non-parallel character and sound sequences. It does not relate words in different languages, thus it cannot encode deciphering constraints similar to the ones considered in this paper. More importantly, this method had not been applied to archaeological data. While lost languages are gaining increasing interest in the NLP community (Knight and Sproat, 2009), there have been no successful attempts of their automatic decipherment.

3 Background on Ugaritic

Manual Decipherment of Ugaritic Ugaritic tablets were first found in Syria in 1929 (Smith, 1955; Watson and Wyatt, 1999). At the time, the cuneiform writing on the tablets was of an unknown type. Charles Virolleaud, who lead the initial decipherment effort, recognized that the script was likely alphabetic, since the inscribed words consisted of only thirty distinct symbols. The location of the tablets discovery further suggested that Ugaritic was likely to have been a Semitic language from the Western branch, with properties similar to Hebrew and Aramaic. This realization was crucial for deciphering the Ugaritic script. In fact, German cryptographer and Semitic scholar Hans Bauer decoded the first two Ugaritic letters—*mem* and *lambda*—by mapping them to Hebrew letters with similar occurrence patterns in prefixes and suffixes. Bootstrapping from this finding, Bauer found words in the tablets that were likely to serve as cognates to Hebrew words—e.g., the Ugaritic word for *king* matches its Hebrew equivalent. Through this process a few more letters were decoded, but the Ugaritic texts were still unreadable. What made the final decipherment possible was a sheer stroke of luck—Bauer guessed that a word inscribed on an ax discovered in the Ras Shamra excavations was the Ugaritic word for *ax*. Bauer's guess was correct, though he selected the wrong phonetic sequence. Edouard Dhorme, another cryptographer

and Semitic scholar, later corrected the reading, expanding a set of translated words. Discoveries of additional tablets allowed Bauer, Dhorme and Virolleaud to revise their hypothesis, successfully completing the decipherment.

Linguistic Features of Ugaritic Ugaritic shares many features with other ancient Semitic languages, following the same word order, gender, number, and case structure (Hetzron, 1997). It is a morphologically rich language, with triliteral roots and many prefixes and suffixes.

At the same time, it exhibits a number of features that distinguish it from Hebrew. Ugaritic has a bigger phonemic inventory than Hebrew, yielding a bigger alphabet – 30 letters vs. 22 in Hebrew. Another distinguishing feature of Ugaritic is that vowels are only written with glottal stops while in Hebrew many long vowels are written using homorganic consonants. Ugaritic also does not have articles, while Hebrew nouns and adjectives take definite articles which are realized as prefixes. These differences result in significant divergence between Hebrew and Ugaritic cognates, thereby complicating the decipherment process.

4 Problem Formulation

We are given a corpus in a lost language and a non-parallel corpus in a related language from the same language family. Our primary goal is to translate words in the unknown language by mapping them to cognates in the known language. As part of this process, we induce a lower-level mapping between the letters of the two alphabets, capturing the regular phonetic correspondences found in cognates.

We make several assumptions about the writing system of the lost language. First, we assume that the writing system is alphabetic in nature. In general, this assumption can be easily validated by counting the number of symbols found in the written record. Next, we assume that the corpus has been transcribed into electronic format, where the graphemes present in the physical text have been unambiguously identified. Finally, we assume that words are explicitly separated in the text, either by white space or a special symbol.

We also make a mild assumption about the morphology of the lost language. We posit that each word consists of a stem, prefix, and suffix, where the latter two may be omitted. This assumption captures a wide range of human languages and a variety of morphological systems. While the correct morphological analysis of words in the lost language must be learned, we assume that the inventory and frequencies of prefixes and suffixes in the known language are given.

In summary, the observed input to the model consists of two elements: (i) a list of unanalyzed word types derived from a corpus in the lost language, and (ii) a morphologically analyzed lexicon in a known related language derived from a separate corpus, in our case non-parallel.

5 Model

5.1 Intuitions

Our goal is to incorporate the logic and intuition used by human decipherers in an unsupervised statistical model. To make these intuitions concrete, consider the following toy example, consisting of a lost language much like English, but written using numerals:

- 15234 *(asked)*
- 1525 *(asks)*
- 4352 *(desk)*

Analyzing the undeciphered corpus, we might first notice a pair of endings, -34, and -5, which both occur after the initial sequence 152- (and may likewise occur at the end of a variety of words in the corpus). If we know this lost language to be closely related to English, we can surmise that these two endings correspond to the English verbal suffixes *-ed* and *-s*. Using this knowledge, we can hypothesize the following character correspondences: (3 = *e*), (4 = *d*), (5 = *s*). We now know that (4252 = *des*2) and we can use our knowledge of the English lexicon to hypothesize that this word is *desk*, thereby learning the correspondence (2 = *k*). Finally, we can use similar reasoning to reveal that the initial character sequence 152- corresponds to the English verb *ask*.

As this example illustrates, human decipherment efforts proceed by discovering both character-level and morpheme-level correspondences. This interplay implicitly relies on a morphological analysis of words in the lost language, while utilizing knowledge of the known language's lexicon and morphology.

One final intuition our model should capture is the sparsity of the alphabetic correspondence between related languages. We know from comparative linguistics that the correct mapping will pre-

serve regular phonetic relationships between the two languages (as exemplified by cognates). As a result, each character in one language will map to a small number of characters in the other language (typically one, but sometimes two or three). By incorporating this structural sparsity intuition, we can allow the model to focus on on a smaller set of linguistically valid hypotheses.

Below we give an overview of our model, which is designed to capture these linguistic intuitions.

5.2 Model Structure

Our model posits that every observed word in the lost language is composed of a sequence of morphemes (prefix, stem, suffix). Furthermore we posit that each morpheme was probabilistically generated jointly with a latent counterpart in the known language.

Our goal is to find those counterparts that lead to high frequency correspondences both at the character and morpheme level. The technical challenge is that each level of correspondence (character and morpheme) can completely describe the observed data. A probabilistic mechanism based simply on one leaves no room for the other to play a role. We resolve this tension by employing a non-parametric Bayesian model: the distributions over bilingual morpheme pairs assign probability based on recurrent patterns at the morpheme level. These distributions are themselves drawn from a prior probabilistic process which favors distributions with consistent character-level correspondences.

We now give a formal description of the model (see Figure 1 for a graphical overview). There are four basic layers in the generative process:

1. **Structural sparsity**: draw a set of indicator variables $\vec{\lambda}$ corresponding to character-edit operations.

2. **Character-edit distribution**: draw a *base distribution* G_0 parameterized by weights on character-edit operations.

3. **Morpheme-pair distributions**: draw a set of distributions on bilingual morpheme pairs $G_{stm}, G_{pre|stm}, G_{suf|stm}$.

4. **Word generation**: draw pairs of cognates in the lost and known language, as well as words in the lost language with no cognate counterpart.

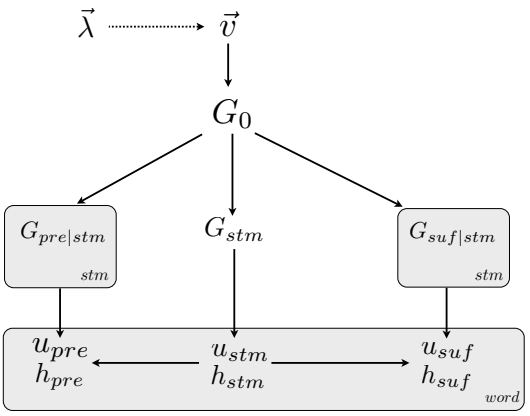


Figure 1: Plate diagram of the decipherment model. The structural sparsity indicator variables $\vec{\lambda}$ determine the values of the base distribution hyperparameters $\vec{v}$. The base distribution G_0 defines probabilities over string-pairs based solely on character-level edits. The morpheme-pair distributions $G_{stm}, G_{pre|stm}, G_{suf|stm}$ directly assign probabilities to highly frequent morpheme pairs.

We now go through each step in more detail.

Structural Sparsity The first step of the generative process provides a control on the sparsity of edit-operation probabilities, encoding the linguistic intuition that the correct character-level mappings should be sparse. The set of edit operations includes character substitutions, insertions, and deletions, as well as a special end symbol: $\{(u,h),(\epsilon,h),(u,\epsilon),END\}$ (where u and h range over characters in the lost and known languages, respectively). For each edit operation e we posit a corresponding indicator variable λ_e. The set of character substitutions with indicators set to one, $\{(u,h) : \lambda_{(u,h)} = 1\}$) conveys the set of phonetically valid correspondences. We define a joint prior over these variables to encourage sparse character mappings. This prior can be viewed as a distribution over *binary matrices* and is defined to encourage rows and columns to sum to low integer values (typically 1). More precisely, for each character u in the lost language, we count the number of mappings $c(u) = \sum_h \lambda_{(u,h)}$. We then define a set of features which count how many of these characters map to i other characters beyond some budget b_i: $f_i = \max(0, |\{u : c(u) = i\}| - b_i)$. Likewise, we define corresponding features f'_i and budgets b'_i for the characters h in the known lan-

guage. The prior over $\vec{\lambda}$ is then defined as

$$P(\vec{\lambda}) = \frac{\exp\left(\vec{f} \cdot \vec{w} + \vec{f}' \cdot \vec{w}\right)}{Z} \quad (1)$$

where the feature weight vector $\vec{w}$ is set to encourage sparse mappings, and Z is a corresponding normalizing constant, which we never need compute. We set $\vec{w}$ so that each character must map to at least one other character, and so that mappings to more than one other character are discouraged [2]

Character-edit Distribution The next step in the generative process is drawing a base distribution G_0 over character edit sequences (each of which yields a bilingual pair of morphemes). This distribution is parameterized by a set of weights $\vec{\phi}$ on edit operations, where the weights over substitutions, insertions, and deletions each individually sum to one. In addition, G_0 provides a fixed distribution q over the *number* of insertions and deletions occurring in any single edit sequence. Probabilities over edit sequences (and consequently on bilingual morpheme pairs) are then defined according to G_0 as:

$$P(\vec{e}) = \prod_i \phi_{e_i} \cdot q\left(\#_{ins}(\vec{e}), \#_{del}(\vec{e})\right)$$

We observe that the average Ugaritic word is over two letters longer than the average Hebrew word. Thus, occurrences of Hebrew character insertions are *a priori* likely, and Ugaritic character deletions are very unlikely. In our experiments, we set q to disallow Ugaritic deletions, and to allow one Hebrew insertion per morpheme (with probability 0.4).

The prior on the base distribution G_0 is a Dirichlet distribution with hyperparameters $\vec{v}$, i.e., $\vec{\phi} \sim Dirichlet(\vec{v})$. Each value v_e thus corresponds to a character edit operation e. Crucially, the value of each v_e depends deterministically on its corresponding indicator variable:

$$v_e = \begin{cases} 1 & \text{if } \lambda_e = 0, \\ K & \text{if } \lambda_e = 1. \end{cases}$$

where K is some constant value > 1.[3] The overall effect is that when $\lambda_e = 0$, the marginal prior density of the corresponding edit weight ϕ_e spikes at 0. When $\lambda_e = 1$, the corresponding marginal prior density remains relatively flat and unconstrained. See (Ishwaran and Rao, 2005) for a similar application of "spike-and-slab" priors in the regression scenario.

Morpheme-pair Distributions Next we draw a series of distributions which *directly* assign probability to morpheme pairs. The previously drawn base distribution G_0 along with a fixed concentration parameter α define a *Dirichlet process* (Antoniak, 1974): $DP(G_0, \alpha)$, which provides probabilities over morpheme-pair distributions. The resulting distributions are likely to be skewed in favor of a few frequently occurring morpheme-pairs, while remaining sensitive to the character-level probabilities of the base distribution.

Our model distinguishes between three types of morphemes: prefixes, stems, and suffixes. As a result, we model each morpheme type as arising from distinct Dirichlet processes, that share a single base distribution:

$$\begin{aligned} G_{stm} &\sim DP(G_0, \alpha_{stm}) \\ G_{pre|stm} &\sim DP(G_0, \alpha_{pre}) \\ G_{suf|stm} &\sim DP(G_0, \alpha_{suf}) \end{aligned}$$

We model prefix and suffix distributions as conditionally dependent on the part-of-speech of the stem morpheme-pair. This choice capture the linguistic fact that different parts-of-speech bear distinct affix frequencies. Thus, while we draw a single distribution G_{stm}, we maintain separate distributions $G_{pre|stm}$ and $G_{suf|stm}$ for each possible stem part-of-speech.

Word Generation Once the morpheme-pair distributions have been drawn, actual word pairs may now be generated. First the model draws a boolean variable c_i to determine whether word i in the lost language has a cognate in the known language, according to some prior $P(c_i)$. If $c_i = 1$, then a cognate word pair (u, h) is produced:

$$\begin{aligned} (u_{stm}, h_{stm}) &\sim G_{stm} \\ (u_{pre}, h_{pre}) &\sim G_{pre|stm} \\ (u_{suf}, h_{suf}) &\sim G_{suf|stm} \\ u &= u_{pre} u_{stm} u_{suf} \\ h &= h_{pre} h_{stm} h_{suf} \end{aligned}$$

Otherwise, a lone word u is generated, according a uniform character-level language model.

[2] We set $w_0 = -\infty, w_1 = 0, w_2 = -50, w_{>2} = -\infty$, with budgets $b'_2 = 7, b'_3 = 1$ (otherwise zero), reflecting the knowledge that there are eight more Ugaritic than Hebrew letters.

[3] Set to 50 in our experiments.

In summary, this model structure captures both character and lexical level correspondences, while utilizing morphological knowledge of the known language. An additional feature of this multi-layered model structure is that each distribution over morpheme pairs is derived from the single character-level base distribution G_0. As a result, any character-level mappings learned from one type of morphological correspondence will be propagated to all other morpheme distributions. Finally, the character-level mappings discovered by the model are encouraged to obey linguistically motivated structural sparsity constraints.

6 Inference

For each word u_i in our undeciphered language we predict a morphological segmentation $(u_{pre}u_{stm}u_{suf})_i$ and corresponding cognate in the known language $(h_{pre}h_{stm}h_{suf})_i$. Ideally we would like to predict the analysis with highest marginal probability under our model given the observed undeciphered corpus and related language lexicon. In order to do so, we need to integrate out all the other latent variables in our model. As these integrals are intractable to compute exactly, we resort to the standard Monte Carlo approximation. We collect samples of the variables over which we wish to marginalize but for which we cannot compute closed-form integrals. We then approximate the marginal probabilities for undeciphered word u_i by summing over all the samples, and predicting the analysis with highest probability.

In our sampling algorithm, we avoid sampling the base distribution G_0 and the derived morpheme-pair distributions (G_{stm} etc.), instead using analytical closed forms. We explicitly sample the sparsity indicator variables $\vec{\lambda}$, the cognate indicator variables c_i, and latent word analyses (segmentations and Hebrew counterparts). To do so tractably, we use Gibbs sampling to draw each latent variable conditioned on our current sample of the others. Although the samples are no longer independent, they form a Markov chain whose stationary distribution is the true joint distribution defined by the model (Geman and Geman, 1984).

6.1 Sampling Word Analyses

For each undeciphered word, we need to sample a morphological segmentation $(u_{pre}, u_{stm}, u_{suf})_i$ along with latent morphemes in the known language $(h_{pre}, h_{stm}, h_{suf})_i$. More precisely, we need to sample three character-edit sequences $\vec{e}_{pre}, \vec{e}_{stm}, \vec{e}_{suf}$ which together yield the observed word u_i.

We break this into two sampling steps. First we sample the morphological segmentation of u_i, along with the part-of-speech pos of the latent stem cognate. To do so, we enumerate each possible segmentation and part-of-speech and calculate its joint conditional probability (for notational clarity, we leave implicit the conditioning on the other samples in the corpus):

$$P(u_{pre}, u_{stm}, u_{suf}, pos) = \sum_{\vec{e}_{stm}} P(\vec{e}_{stm}) \sum_{\vec{e}_{pre}} P(\vec{e}_{pre}|pos) \sum_{\vec{e}_{suf}} P(\vec{e}_{suf}|pos) \quad (2)$$

where the summations over character-edit sequences are restricted to those which yield the segmentation $(u_{pre}, u_{stm}, u_{suf})$ and a latent cognate with part-of-speech pos.

For a particular stem edit-sequence $\vec{e}_{stm}$, we compute its conditional probability in closed form according to a Chinese Restaurant Process (Antoniak, 1974). To do so, we use counts from the other sampled word analyses: $count_{stm}(\vec{e}_{stm})$ gives the number of times that the entire edit-sequence $\vec{e}_{stm}$ has been observed:

$$P(\vec{e}_{stm}) \propto \frac{count_{stm}(\vec{e}_{stm}) + \alpha \prod_i p(e_i)}{n + \alpha}$$

where n is the number of other word analyses sampled, and α is a fixed concentration parameter. The product $\prod_i p(e_i)$ gives the probability of $\vec{e}_{stm}$ according to the *base distribution* G_0. Since the parameters of G_0 are left unsampled, we use the marginalized form:

$$p(e) = \frac{v_e + count(e)}{\sum_{e'} v_{e'} + k} \quad (3)$$

where $count(e)$ is the number of times that character-edit e appears in distinct edit-sequences (across prefixes, stems, and suffixes), and k is the sum of these counts across all character-edits. Recall that v_e is a hyperparameter for the Dirichlet prior on G_0 and depends on the value of the corresponding indicator variable λ_e.

Once the segmentation $(u_{pre}, u_{stm}, u_{suf})$ and part-of-speech pos have been sampled, we proceed to sample the actual edit-sequences (and thus

latent morphemes counterparts). Now, instead of summing over the values in Equation 2, we instead sample from them.

6.2 Sampling Sparsity Indicators

Recall that each sparsity indicator λ_e determines the value of the corresponding hyperparameter v_e of the Dirichlet prior for the character-edit base distribution G_0. In addition, we have an unnormalized joint prior $P(\vec{\lambda}) = \frac{g(\vec{\lambda})}{Z}$ which encourages a sparse setting of these variables. To sample a particular λ_e, we consider the set $\vec{\lambda}$ in which $\lambda_e = 0$ and $\vec{\lambda}'$ in which $\lambda_e = 1$. We then compute:

$$P(\vec{\lambda}) \propto g(\vec{\lambda}) \cdot \frac{v_e^{[count(e)]}}{\sum_{e'} v_{e'}^{[k]}}$$

where k is the sum of counts for all edit operations, and the notation $a^{[b]}$ indicates the ascending factorial. Likewise, we can compute a probability for $\vec{\lambda}'$ with corresponding values v'_e.

6.3 Sampling Cognate Indicators

Finally, for each word u_i, we sample a corresponding indicator variable c_i. To do so, we calculate Equation 2 for all possible segmentations and parts-of-speech and sum the resulting values to obtain the conditional likelihood $P(u_i|c_i = 1)$. We also calculate $P(u_i|c_i = 0)$ using a uniform unigram character-level language model (and thus depends only on the number of characters in u_i). We then sample from among the two values:

$$P(u_i|c_i = 1) \cdot P(c_i = 1)$$
$$P(u_i|c_i = 0) \cdot P(c_i = 0)$$

6.4 High-level Resampling

Besides the individual sampling steps detailed above, we also consider several larger sampling moves in order to speed convergence. For example, for each type of edit-sequence $\vec{e}$ which has been sampled (and may now occur many times throughout the data), we consider a single joint move to another edit-sequence $\vec{e'}$ (both of which yield the same lost language morpheme u). The details are much the same as above, and as before the set of possible edit-sequences is limited by the string u and the known language lexicon.

We also resample groups of the sparsity indicator variables $\vec{\lambda}$ in tandem, to allow a more rapid exploration of the probability space. For each character u, we block sample the entire set $\{\lambda_{(u,h)}\}_h$, and likewise for each character h.

6.5 Implementation Details

Many of the steps detailed above involve the consideration of all possible edit-sequences consistent with (i) a particular undeciphered word u_i and (ii) the entire lexicon of words in the known language (or some subset of words with a particular part-of-speech). In particular, we need to both sample from and sum over this space of possibilities repeatedly. Doing so by simple enumeration would needlessly repeat many sub-computations. Instead we use finite-state acceptors to compactly represent both the entire Hebrew lexicon as well as potential Hebrew word forms for each Ugaritic word. By intersecting two such FSAs and minimizing the result we can efficiently represent all potential Hebrew words for a particular Ugaritic word. We weight the edges in the FSA according to the base distribution probabilities (in Equation 3 above). Although these intersected acceptors have to be constantly reweighted to reflect changing probabilities, their topologies need only be computed once. One weighted correctly, marginals and samples can be computed using dynamic programming.

Even with a large number of sampling rounds, it is difficult to fully explore the latent variable space for complex unsupervised models. Thus a clever initialization is usually required to start the sampler in a high probability region. We initialize our model with the results of the HMM-based baseline (see section 8), and rule out character substitutions with probability < 0.05 according to the baseline.

7 Experiments

7.1 Corpus and Annotations

We apply our model to the ancient Ugaritic language (see Section 3 for background). Our undeciphered corpus consists of an electronic transcription of the Ugaritic tablets (Cunchillos et al., 2002). This corpus contains 7,386 unique word types. As our known language corpus, we use the Hebrew Bible, which is both geographically and temporally close to Ugaritic. To extract a Hebrew morphological lexicon we assume the existence of manual morphological and part-of-speech annotations (Groves and Lowery, 2006). We divide Hebrew stems into four main part-of-speech categories each with a distinct affix profile: Noun, Verb, Pronoun, and Particle. For each part-of-speech category, we determine the set of allowable affixes using the annotated Bible corpus.

	Words		Morphemes	
	type	token	type	token
Baseline	28.82%	46.00%	N/A	N/A
Our Model	60.42%	66.71%	75.07%	81.25%
No Sparsity	46.08%	54.01%	69.48%	76.10%

Table 1: Accuracy of cognate translations, measured with respect to complete word-forms and morphemes, for the HMM-based substitution cipher baseline, our complete model, and our model without the structural sparsity priors. Note that the baseline does not provide per-morpheme results, as it does not predict morpheme boundaries.

To evaluate the output of our model, we annotated the words in the Ugaritic lexicon with the corresponding Hebrew cognates found in the standard reference dictionary (del Olo Lete and Sanmartín, 2004). In addition, manual morphological segmentation was carried out with the guidance of a standard Ugaritic grammar (Schniedewind and Hunt, 2007). Although Ugaritic is an inflectional rather than agglutinative language, in its written form (which lacks vowels) words can easily be segmented (e.g. *wyplṭn* becomes *wy-plṭ-n*).

Overall, we identified Hebrew cognates for 2,155 word forms, covering almost 1/3 of the Ugaritic vocabulary.[4]

8 Evaluation Tasks and Results

We evaluate our model on four separate decipherment tasks: (i) Learning alphabetic mappings, (ii) translating cognates, (iii) identifying cognates, and (iv) morphological segmentation.

As a baseline for the first three of these tasks (learning alphabetic mappings and translating and identifying cognates), we adapt the HMM-based method of Knight et al. (2006) for learning letter substitution ciphers. In its original setting, this model was used to map written texts to spoken language, under the assumption that each character was emitted from a hidden phonemic state. In our adaptation, we assume instead that each Ugaritic character was generated by a hidden Hebrew letter. Hebrew character trigram transition probabilities are estimated using the Hebrew Bible, and Hebrew to Ugaritic character emission probabilities are learned using EM. Finally, the highest probability sequence of latent Hebrew letters is predicted for each Ugaritic word-form, using Viterbi decoding.

Alphabetic Mapping The first essential step towards successful decipherment is recovering the mapping between the symbols of the lost language and the alphabet of a known language. As a gold standard for this comparison, we use the well-established relationship between the Ugaritic and Hebrew alphabets (Hetzron, 1997). This mapping is not one-to-one but is generally quite sparse. Of the 30 Ugaritic symbols, 28 map predominantly to a single Hebrew letter, and the remaining two map to two different letters. As the Hebrew alphabet contains only 22 letters, six map to two distinct Ugaritic letters and two map to three distinct Ugaritic letters.

We recover our model's predicted alphabetic mappings by simply examining the sampled values of the binary indicator variables $\lambda_{u,h}$ for each Ugaritic-Hebrew letter pair (u, h). Due to our structural sparsity prior $P(\vec{\lambda})$, the predicted mappings are sparse: each Ugaritic letter maps to only a single Hebrew letter, and most Hebrew letters map to only a single Ugaritic letter. To recover alphabetic mappings from the HMM substitution cipher baseline, we predict the Hebrew letter h which maximizes the model's probability $P(h|u)$, for each Ugaritic letter u.

To evaluate these mappings, we simply count the number of Ugaritic letters that are correctly mapped to one of their Hebrew reflexes. By this measure, the baseline recovers correct mappings for 22 out of 30 Ugaritic characters (73.3%). Our model recovers correct mappings for all but one (very low frequency) Ugaritic characters, yielding 96.67% accuracy.

Cognate Decipherment We compare the decipherment accuracy for Ugaritic words that have corresponding Hebrew cognates. We evaluate our model's predictions on each distinct Ugaritic word-form at both the type and token level. As Table 1 shows, our method correctly translates over 60% of all distinct Ugaritic word-forms with Hebrew cognates and over 71% of the individual morphemes that compose them, outperforming the baseline by significant margins. Accuracy improves when the frequency of the word-forms is taken into account (token-level evaluation), indicating that the model is able to decipher frequent words more accurately than infre-

[4] We are confident that a large majority of Ugaritic words with known Hebrew cognates were thus identified. The remaining Ugaritic words include many personal and geographic names, words with cognates in other Semitic languages, and words whose etymology is uncertain.

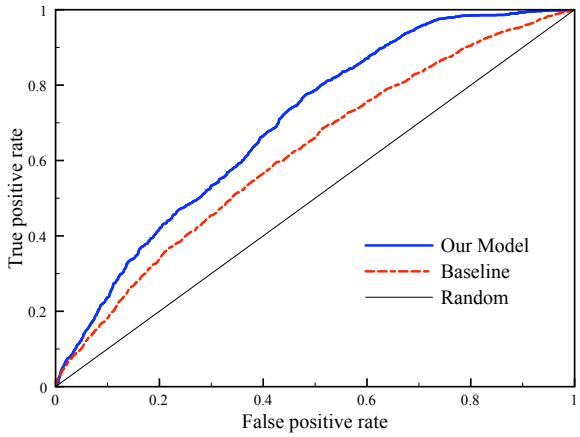


Figure 2: ROC curve for cognate identification.

quent words. We also measure the average Levenshtein distance between predicted and actual cognate word-forms. On average, our model's predictions lie 0.52 edit operations from the true cognate, whereas the baseline's predictions average a distance of 1.26 edit operations.

Finally, we evaluated the performance of our model when the structural sparsity constraints are not used. As Table 1 shows, performance degrades significantly in the absence of these priors, indicating the importance of modeling the sparsity of character mappings.

Cognate identification We evaluate our model's ability to identify cognates using the sampled indicator variables c_i. As before, we compare our performance against the HMM substitution cipher baseline. To produce baseline cognate identification predictions, we calculate the probability of each latent Hebrew letter sequence predicted by the HMM, and compare it to a uniform character-level Ugaritic language model (as done by our model, to avoid automatically assigning higher cognate probability to shorter Ugaritic words). For both our model and the baseline, we can vary the threshold for cognate identification by raising or lowering the cognate prior $P(c_i)$. As the prior is set higher, we detect more true cognates, but the false positive rate increases as well.

Figure 2 shows the ROC curve obtained by varying this prior both for our model and the baseline. At all operating points, our model outperforms the baseline, and both models always predict better than chance. In practice for our model, we use a high cognate prior, thus only ruling out those Ugaritic word-forms which are very unlikely to have Hebrew cognates.

	precision	recall	f-measure
Morfessor	88.87%	67.48%	76.71%
Our Model	86.62%	90.53%	88.53%

Table 2: Morphological segmentation accuracy for a standard unsupervised baseline and our model.

Morphological segmentation Finally, we evaluate the accuracy of our model's morphological segmentation for Ugaritic words. As a baseline for this comparison, we use Morfessor Categories-MAP (Creutz and Lagus, 2007). As Table 2 shows, our model provides a significant boost in performance, especially for recall. This result is consistent with previous work showing that morphological annotations can be projected to new languages lacking annotation (Yarowsky et al., 2000; Snyder and Barzilay, 2008), but generalizes those results to the case where parallel data is unavailable.

9 Conclusion and Future Work

In this paper we proposed a method for the automatic decipherment of lost languages. The key strength of our model lies in its ability to incorporate a range of linguistic intuitions in a statistical framework.

We hope to address several issues in future work. Our model fails to take into account the known *frequency* of Hebrew words and morphemes. In fact, the most common error is incorrectly translating the masculine plural suffix (*-m*) as the third person plural possessive suffix (*-m*) rather than the correct and much more common plural suffix (*-ym*). Also, even with the correct alphabetic mapping, many words can only be deciphered by examining their literary context. Our model currently operates purely on the vocabulary level and thus fails to take this contextual information into account. Finally, we intend to explore our model's predictive power when the family of the lost language is unknown.[5]

[5]The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835445, and grant IIS-0835652) and the Microsoft Research New Faculty Fellowship. Thanks to Michael Collins, Tommi Jaakkola, and the MIT NLP group for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, November.

Alexandre Bouchard, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A probabilistic approach to diachronic phonology. In *Proceedings of EMNLP*, pages 887–896.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Jesus-Luis Cunchillos, Juan-Pablo Vita, and Jose-Ángel Zamora. 2002. Ugaritic data bank. CD-ROM.

Gregoria del Olo Lete and Joaquín Sanmartín. 2004. *A Dictionary of the Ugaritic Language in the Alphabetic Tradition*. Number 67 in Handbook of Oriental Studies. Section 1 The Near and Middle East. Brill.

Pascale Fung and Kathleen McKeown. 1997. Finding terminology translations from non-parallel corpora. In *Proceedings of the Annual Workshop on Very Large Corpora*, pages 192–202.

S. Geman and D. Geman. 1984. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:609–628.

Alan Groves and Kirk Lowery, editors. 2006. *The Westminster Hebrew Bible Morphology Database*. Westminster Hebrew Institute, Philadelphia, PA, USA.

Jacques B. M. Guy. 1994. An algorithm for identifying cognates in bilingual wordlists and its applicability to machine translation. *Journal of Quantitative Linguistics*, 1(1):35–42.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the ACL/HLT*, pages 771–779.

Robert Hetzron, editor. 1997. *The Semitic Languages*. Routledge.

H. Ishwaran and J.S. Rao. 2005. Spike and slab variable selection: frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2):730–773.

Kevin Knight and Richard Sproat. 2009. Writing systems, transliteration and decipherment. NAACL Tutorial.

K. Knight and K. Yamada. 1999. A computational approach to deciphering unknown scripts. In *ACL Workshop on Unsupervised Learning in Natural Language Processing*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL*, pages 499–506.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition*, pages 9–16.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceeding of NAACL*, pages 1–8.

Grzegorz Kondrak. 2009. Identification of cognates and recurrent sound correspondences in word lists. *Traitement Automatique des Langues*, 50(2):201–235.

John B. Lowe and Martine Mazaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistics*, 20(3):381–417.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the ACL*, pages 519–526.

Andrew Robinson. 2002. *Lost Languages: The Enigma of the World's Undeciphered Scripts*. McGraw-Hill.

William M. Schniedewind and Joel H. Hunt. 2007. *A Primer on Ugaritic: Language, Culture and Literature*. Cambridge University Press.

Mark S. Smith, editor. 1955. *Untold Stories: The Bible and Ugaritic Studies in the Twentieth Century*. Hendrickson Publishers.

Benjamin Snyder and Regina Barzilay. 2008. Cross-lingual propagation for morphological analysis. In *Proceedings of the AAAI*, pages 848–854.

Wilfred Watson and Nicolas Wyatt, editors. 1999. *Handbook of Ugaritic Studies*. Brill.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2000. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*, pages 161–168.

Efficient Inference Through Cascades of Weighted Tree Transducers

Jonathan May and Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{jonmay,knight}@isi.edu

Heiko Vogler
Technische Universität Dresden
Institut für Theoretische Informatik
01062 Dresden, Germany
heiko.vogler@tu-dresden.de

Abstract

Weighted tree transducers have been proposed as useful formal models for representing syntactic natural language processing applications, but there has been little description of inference algorithms for these automata beyond formal foundations. We give a detailed description of algorithms for application of cascades of weighted tree transducers to weighted tree acceptors, connecting formal theory with actual practice. Additionally, we present novel on-the-fly variants of these algorithms, and compare their performance on a syntax machine translation cascade based on (Yamada and Knight, 2001).

1 Motivation

Weighted finite-state transducers have found recent favor as models of natural language (Mohri, 1997). In order to make actual use of systems built with these formalisms we must first calculate the set of possible weighted outputs allowed by the transducer given some input, which we call *forward application*, or the set of possible weighted inputs given some output, which we call *backward application*. After application we can do some inference on this result, such as determining its k highest weighted elements.

We may also want to divide up our problems into manageable chunks, each represented by a transducer. As noted by Woods (1980), it is easier for designers to write several small transducers where each performs a simple transformation, rather than painstakingly construct a single complicated device. We would like to know, then, the result of transformation of input or output by a *cascade* of transducers, one operating after the other. As we will see, there are various strategies for approaching this problem. We will consider *offline composition*, *bucket brigade application*, and *on-the-fly application*.

Application of cascades of weighted string transducers (WSTs) has been well-studied (Mohri, 1997). Less well-studied but of more recent interest is application of cascades of weighted *tree* transducers (WTTs). We tackle application of WTT cascades in this work, presenting:

- explicit algorithms for application of WTT cascades
- novel algorithms for on-the-fly application of WTT cascades, and
- experiments comparing the performance of these algorithms.

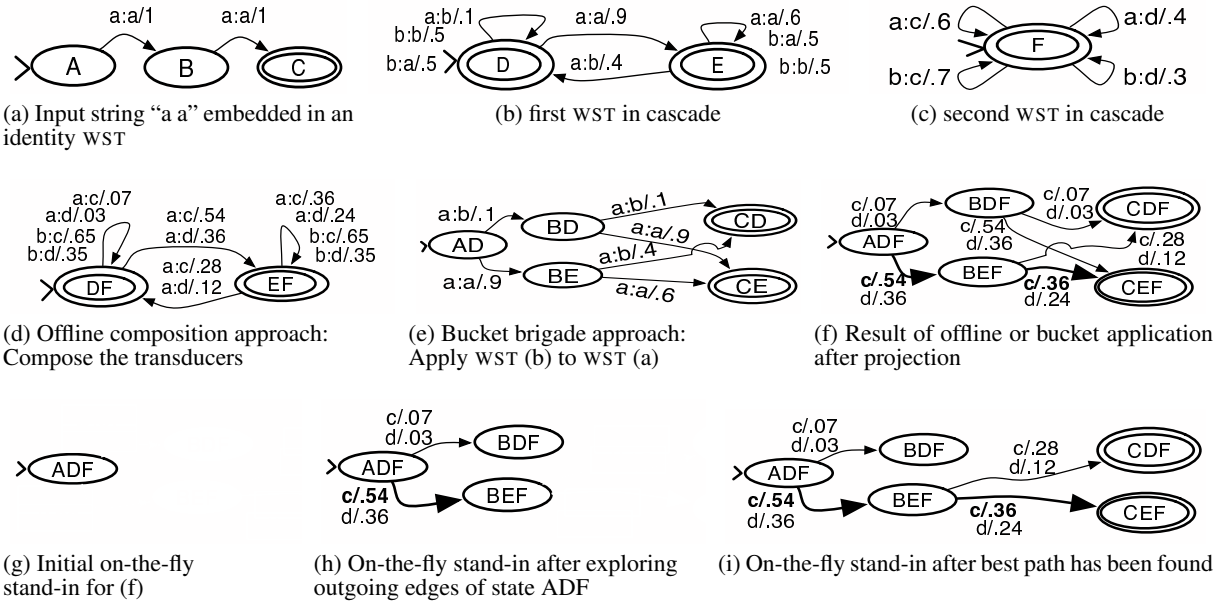
2 Strategies for the string case

Before we discuss application of WTTs, it is helpful to recall the solution to this problem in the WST domain. We recall previous formal presentations of WSTs (Mohri, 1997) and note informally that they may be represented as directed graphs with designated start and end states and edges labeled with input symbols, output symbols, and weights.[1] Fortunately, the solution for WSTs is practically trivial—we achieve application through a series of *embedding*, *composition*, and *projection* operations. Embedding is simply the act of representing a string or regular string language as an identity WST. Composition of WSTs, that is, generating a single WST that captures the transformations of two input WSTs used in sequence, is not at all trivial, but has been well covered in, e.g., (Mohri, 2009), where directly implementable algorithms can be found. Finally, projection is another trivial operation—the domain or range language can be obtained from a WST by ignoring the output or input symbols, respectively, on its arcs, and summing weights on otherwise identical arcs. By embedding an input, composing the result with the given WST, and projecting the result, forward application is accomplished.[2] We are then left with a weighted string acceptor (WSA), essentially a weighted, labeled graph, which can be traversed

[1] We assume throughout this paper that weights are in $\mathbb{R}_+ \cup \{+\infty\}$, that the weight of a path is calculated as the product of the weights of its edges, and that the weight of a (not necessarily finite) set T of paths is calculated as the sum of the weights of the paths of T.

[2] For backward applications, the roles of input and output are simply exchanged.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1058–1066,
Uppsala, Sweden, 11-16 July 2010.



(a) Input string "a a" embedded in an identity WST

(b) first WST in cascade

(c) second WST in cascade

(d) Offline composition approach: Compose the transducers

(e) Bucket brigade approach: Apply WST (b) to WST (a)

(f) Result of offline or bucket application after projection

(g) Initial on-the-fly stand-in for (f)

(h) On-the-fly stand-in after exploring outgoing edges of state ADF

(i) On-the-fly stand-in after best path has been found

Figure 1: Three different approaches to application through cascades of WSTs.

by well-known algorithms to efficiently find the k-best paths.

Because WSTs can be freely composed, extending application to operate on a *cascade* of WSTs is fairly trivial. The only question is one of composition order: whether to initially compose the cascade into a single transducer (an approach we call *offline composition*) or to compose the initial embedding with the first transducer, trim useless states, compose the result with the second, and so on (an approach we call *bucket brigade*). The appropriate strategy generally depends on the structure of the individual transducers.

A third approach builds the result incrementally, as dictated by some algorithm that requests information about it. Such an approach, which we call *on-the-fly*, was described in (Pereira and Riley, 1997; Mohri, 2009; Mohri et al., 2000). If we can efficiently calculate the outgoing edges of a state of the result WSA on demand, without calculating all edges in the entire machine, we can maintain a *stand-in* for the result structure, a machine consisting at first of only the start state of the true result. As a calling algorithm (e.g., an implementation of Dijkstra's algorithm) requests information about the result graph, such as the set of outgoing edges from a state, we replace the current stand-in with a richer version by adding the result of the request. The on-the-fly approach has a distinct advantage over the other two methods in that the entire result graph need not be built. A graphical representation of all three methods is presented in Figure 1.

3 Application of tree transducers

Now let us revisit these strategies in the setting of trees and tree transducers. Imagine we have a tree or set of trees as input that can be represented as a weighted regular tree grammar[3] (WRTG) and a WTT that can transform that input with some weight. We would like to know the k-best trees the WTT can produce as output for that input, along with their weights. We already know of several methods for acquiring k-best trees from a WRTG (Huang and Chiang, 2005; Pauls and Klein, 2009), so we then must ask if, analogously to the string case, WTTs preserve recognizability[4] and we can form an application WRTG. Before we begin, however, we must define WTTs and WRTGs.

3.1 Preliminaries[5]

A *ranked alphabet* is a finite set Σ such that every member $\sigma \in \Sigma$ has a rank $rk(\sigma) \in \mathbb{N}$. We call $\Sigma^{(k)} \subseteq \Sigma, k \in \mathbb{N}$ the set of those $\sigma \in \Sigma$ such that $rk(\sigma) = k$. The set of *variables* is denoted $X = \{x_1, x_2, \ldots\}$ and is assumed to be disjoint from any ranked alphabet used in this paper. We use $\perp$ to denote a symbol of rank 0 that is not in any ranked alphabet used in this paper. A *tree* $t \in T_\Sigma$ is denoted $\sigma(t_1, \ldots, t_k)$ where $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T_\Sigma$. For $\sigma \in \Sigma^{(0)}$ we

[3] This generates the same class of weighted tree languages as weighted tree automata, the direct analogue of WSAs, and is more useful for our purposes.

[4] A weighted tree language is recognizable iff it can be represented by a wrtg.

[5] The following formal definitions and notations are needed for understanding and reimplementation of the presented algorithms, but can be safely skipped on first reading and consulted when encountering an unfamiliar term.

write $\sigma \in T_\Sigma$ as shorthand for $\sigma()$. For every set S disjoint from Σ, let $T_\Sigma(S) = T_{\Sigma \cup S}$, where, for all $s \in S$, $rk(s) = 0$.

We define the *positions* of a tree $t = \sigma(t_1, \ldots, t_k)$, for $k \geq 0$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma$, as a set $pos(t) \subset \mathbb{N}^*$ such that $pos(t) = \{\varepsilon\} \cup \{iv \mid 1 \leq i \leq k, v \in pos(t_i)\}$. The set of leaf positions $lv(t) \subseteq pos(t)$ are those positions $v \in pos(t)$ such that for no $i \in \mathbb{N}$, $vi \in pos(t)$. We presume standard lexicographic orderings $<$ and $\leq$ on pos.

Let $t, s \in T_\Sigma$ and $v \in pos(t)$. The *label* of t at position v, denoted by $t(v)$, the *subtree* of t at v, denoted by $t|_v$, and the *replacement* at v by s, denoted by $t[s]_v$, are defined as follows:

1. For every $\sigma \in \Sigma^{(0)}$, $\sigma(\varepsilon) = \sigma$, $\sigma|_\varepsilon = \sigma$, and $\sigma[s]_\varepsilon = s$.
2. For every $t = \sigma(t_1, \ldots, t_k)$ such that $k = rk(\sigma)$ and $k \geq 1$, $t(\varepsilon) = \sigma$, $t|_\varepsilon = t$, and $t[s]_\varepsilon = s$. For every $1 \leq i \leq k$ and $v \in pos(t_i)$, $t(iv) = t_i(v)$, $t|_{iv} = t_i|_v$, and $t[s]_{iv} = \sigma(t_1, \ldots, t_{i-1}, t_i[s]_v, t_{i+1}, \ldots, t_k)$.

The *size* of a tree t, $size(t)$ is $|pos(t)|$, the cardinality of its position set. The *yield set* of a tree is the set of labels of its leaves: for a tree t, $yd(t) = \{t(v) \mid v \in lv(t)\}$.

Let A and B be sets. Let $\varphi : A \to T_\Sigma(B)$ be a mapping. We extend φ to the mapping $\overline{\varphi} : T_\Sigma(A) \to T_\Sigma(B)$ such that for $a \in A$, $\overline{\varphi}(a) = \varphi(a)$ and for $k \geq 0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T_\Sigma(A)$, $\overline{\varphi}(\sigma(t_1, \ldots, t_k)) = \sigma(\overline{\varphi}(t_1), \ldots, \overline{\varphi}(t_k))$. We indicate such extensions by describing φ as a *substitution mapping* and then using $\overline{\varphi}$ without further comment.

We use $\mathbb{R}_+$ to denote the set $\{w \in \mathbb{R} \mid w \geq 0\}$ and $\mathbb{R}_+^\infty$ to denote $\mathbb{R}_+ \cup \{+\infty\}$.

Definition 3.1 (cf. (Alexandrakis and Bozapalidis, 1987)) A weighted regular tree grammar (WRTG) is a 4-tuple $G = (N, \Sigma, P, n_0)$ where:

1. N is a finite set of *nonterminals*, with $n_0 \in N$ the start nonterminal.
2. Σ is a ranked alphabet of input symbols, where $\Sigma \cap N = \emptyset$.
3. P is a tuple (P', π), where P' is a finite set of *productions*, each production p of the form $n \to u$, $n \in N$, $u \in T_\Sigma(N)$, and $\pi : P' \to \mathbb{R}_+$ is a weight function of the productions. We will refer to P as a finite set of weighted productions, each production p of the form $n \xrightarrow{\pi(p)} u$.

A production p is a *chain production* if it is of the form $n_i \xrightarrow{w} n_j$, where $n_i, n_j \in N$.[6]

[6] In (Alexandrakis and Bozapalidis, 1987), chain productions are forbidden in order to avoid infinite summations. We explicitly allow such summations.

A WRTG G is in *normal form* if each production is either a chain production or is of the form $n \xrightarrow{w} \sigma(n_1, \ldots, n_k)$ where $\sigma \in \Sigma^{(k)}$ and $n_1, \ldots, n_k \in N$.

For WRTG $G = (N, \Sigma, P, n_0)$, $s, t, u \in T_\Sigma(N)$, $n \in N$, and $p \in P$ of the form $n \xrightarrow{w} u$, we obtain a *derivation step* from s to t by replacing some leaf nonterminal in s labeled n with u. Formally, $s \Rightarrow_G^p t$ if there exists some $v \in lv(s)$ such that $s(v) = n$ and $s[u]_v = t$. We say this derivation step is *leftmost* if, for all $v' \in lv(s)$ where $v' < v$, $s(v') \in \Sigma$. We henceforth assume all derivation steps are leftmost. If, for some $m \in \mathbb{N}$, $p_i \in P$, and $t_i \in T_\Sigma(N)$ for all $1 \leq i \leq m$, $n_0 \Rightarrow^{p_1} t_1 \ldots \Rightarrow^{p_m} t_m$, we say the sequence $d = (p_1, \ldots, p_m)$ is a *derivation* of t_m in G and that $n_0 \Rightarrow^* t_m$; the weight of d is $wt(d) = \pi(p_1) \cdot \ldots \cdot \pi(p_m)$. The weighted tree language recognized by G is the mapping $L_G : T_\Sigma \to \mathbb{R}_+^\infty$ such that for every $t \in T_\Sigma$, $L_G(t)$ is the sum of the weights of all (possibly infinitely many) derivations of t in G. A weighted tree language $f : T_\Sigma \to \mathbb{R}_+^\infty$ is *recognizable* if there is a WRTG G such that $f = L_G$.

We define a partial ordering $\preceq$ on WRTGs such that for WRTGs $G_1 = (N_1, \Sigma, P_1, n_0)$ and $G_2 = (N_2, \Sigma, P_2, n_0)$, we say $G_1 \preceq G_2$ iff $N_1 \subseteq N_2$ and $P_1 \subseteq P_2$, where the weights are preserved.

Definition 3.2 (cf. Def. 1 of (Maletti, 2008)) A weighted extended top-down tree transducer (WXTT) is a 5-tuple $M = (Q, \Sigma, \Delta, R, q_0)$ where:

1. Q is a finite set of states.
2. Σ and Δ are the ranked alphabets of input and output symbols, respectively, where $(\Sigma \cup \Delta) \cap Q = \emptyset$.
3. R is a tuple (R', π), where R' is a finite set of *rules*, each rule r of the form $q.y \to u$ for $q \in Q$, $y \in T_\Sigma(X)$, and $u \in T_\Delta(Q \times X)$. We further require that no variable $x \in X$ appears more than once in y, and that each variable appearing in u is also in y. Moreover, $\pi : R' \to \mathbb{R}_+^\infty$ is a weight function of the rules. As for WRTGs, we refer to R as a finite set of weighted rules, each rule r of the form $q.y \xrightarrow{\pi(r)} u$.

A WXTT is *linear* (respectively, *nondeleting*) if, for each rule r of the form $q.y \xrightarrow{w} u$, each $x \in yd(y) \cap X$ appears at most once (respectively, at least once) in u. We denote the class of all WXTTs as wxT and add the letters L and N to signify the subclasses of linear and nondeleting WTT, respectively. Additionally, if y is of the form $\sigma(x_1, \ldots, x_k)$, we remove the letter "x" to signify

the transducer is not extended (i.e., it is a "traditional" WTT (Fülöp and Vogler, 2009)).

For WXTT $M = (Q, \Sigma, \Delta, R, q_0)$, $s, t \in T_\Delta(Q \times T_\Sigma)$, and $r \in R$ of the form $q.y \xrightarrow{w} u$, we obtain a *derivation step* from s to t by replacing some leaf of s labeled with q and a tree matching y by a transformation of u, where each instance of a variable has been replaced by a corresponding subtree of the y-matching tree. Formally, $s \Rightarrow_M^r t$ if there is a position $v \in pos(s)$, a substitution mapping $\varphi : X \to T_\Sigma$, and a rule $q.y \xrightarrow{w} u \in R$ such that $s(v) = (q, \overline{\varphi}(y))$ and $t = s[\overline{\varphi'}(u)]_v$, where φ' is a substitution mapping $Q \times X \to T_\Delta(Q \times T_\Sigma)$ defined such that $\varphi'(q', x) = (q', \varphi(x))$ for all $q' \in Q$ and $x \in X$. We say this derivation step is *leftmost* if, for all $v' \in lv(s)$ where $v' < v$, $s(v') \in \Delta$. We henceforth assume all derivation steps are leftmost. If, for some $s \in T_\Sigma$, $m \in \mathbb{N}$, $r_i \in R$, and $t_i \in T_\Delta(Q \times T_\Sigma)$ for all $1 \leq i \leq m$, $(q_0, s) \Rightarrow^{r_1} t_1 \ldots \Rightarrow^{r_m} t_m$, we say the sequence $d = (r_1, \ldots, r_m)$ is a *derivation* of (s, t_m) in M; the weight of d is $wt(d) = \pi(r_1) \cdot \ldots \cdot \pi(r_m)$. The *weighted tree transformation* recognized by M is the mapping $\tau_M : T_\Sigma \times T_\Delta \to \mathbb{R}_+^\infty$, such that for every $s \in T_\Sigma$ and $t \in T_\Delta$, $\tau_M(s, t)$ is the sum of the weights of all (possibly infinitely many) derivations of (s, t) in M. The *composition* of two weighted tree transformations $\tau : T_\Sigma \times T_\Delta \to \mathbb{R}_+^\infty$ and $\mu : T_\Delta \times T_\Gamma \to \mathbb{R}_+^\infty$ is the weighted tree transformation $(\tau; \mu) : T_\Sigma \times T_\Gamma \to \mathbb{R}_+^\infty$ where for every $s \in T_\Sigma$ and $u \in T_\Gamma$, $(\tau; \mu)(s, u) = \sum_{t \in T_\Delta} \tau(s, t) \cdot \mu(t, u)$.

3.2 Applicable classes

We now consider transducer classes where recognizability is preserved under application. Table 1 presents known results for the top-down tree transducer classes described in Section 3.1. Unlike the string case, preservation of recognizability is not universal or symmetric. This is important for us, because we can only construct an *application* WRTG, i.e., a WRTG representing the result of application, if we can ensure that the language generated by application is in fact recognizable. Of the types under consideration, only wxLNT and wLNT preserve forward recognizability. The two classes marked as open questions and the other classes, which are superclasses of wNT, do not or are presumed not to. All subclasses of wxLT preserve backward recognizability.[7] We do not consider cases where recognizability is not preserved in the remainder of this paper. If a transducer M of a class that preserves forward recognizability is applied to a WRTG G, we can call the forward application WRTG $M(G)^\triangleright$ and if M preserves backward recognizability, we can call the backward application WRTG $M(G)^\triangleleft$.

Now that we have explained the application problem in the context of weighted tree transducers and determined the classes for which application is possible, let us consider how to build forward and backward application WRTGs. Our basic approach mimics that taken for WSTs by using an embed-compose-project strategy. As in string world, if we can embed the input in a transducer, compose with the given transducer, and project the result, we can obtain the application WRTG. Embedding a WRTG in a wLNT is a trivial operation—if the WRTG is in normal form and chain production-free,[8] for every production of the form $n \xrightarrow{w} \sigma(n_1, \ldots, n_k)$, create a rule of the form $n.\sigma(x_1, \ldots, x_k) \xrightarrow{w} \sigma(n_1.x_1, \ldots, n_k.x_k)$. Range projection of a wxLNT is also trivial—for every $q \in Q$ and $u \in T_\Delta(Q \times X)$ create a production of the form $q \xrightarrow{w} u'$ where u' is formed from u by replacing all leaves of the form $q.x$ with the leaf q, i.e., removing references to variables, and w is the sum of the weights of all rules of the form $q.y \to u$ in R.[9] Domain projection for wxLT is best explained by way of example. The left side of a rule is preserved, with variables leaves replaced by their associated states from the right side. So, the rule $q_1.\sigma(\gamma(x_1), x_2) \xrightarrow{w} \delta(q_2.x_2, \beta(\alpha, q_3.x_1))$ would yield the production $q_1 \xrightarrow{w} \sigma(\gamma(q_3), q_2)$ in the domain projection. However, a deleting rule such as $q_1.\sigma(x_1, x_2) \xrightarrow{w} \gamma(q_2.x_2)$ necessitates the introduction of a new nonterminal $\perp$ that can generate all of T_Σ with weight 1.

The only missing piece in our embed-compose-project strategy is composition. Algorithm 1, which is based on the declarative construction of Maletti (2006), generates the syntactic composition of a wxLT and a wLNT, a generalization of the basic composition construction of Baker (1979). It calls Algorithm 2, which determines the sequences of rules in the second transducer that match the right side of a single rule in the first transducer. Since the embedded WRTG is of type wLNT, it may be either the first or second argument provided to Algorithm 1, depending on whether the application is forward or backward. We can thus use the embed-compose-project strategy for forward application of wLNT and backward application of wxLT and wxLNT. Note that we *cannot* use this strategy for forward applica-

[7] Note that the introduction of weights limits recognizability preservation considerably. For example, (unweighted) xT preserves backward recognizability.

[8] Without loss of generality we assume this is so, since standard algorithms exist to remove chain productions (Kuich, 1998; Ésik and Kuich, 2003; Mohri, 2009) and convert into normal form (Alexandrakis and Bozapalidis, 1987).

[9] Finitely many such productions may be formed.

tion of wxLNT, even though that class preserves recognizability.

Algorithm 1 COMPOSE

inputs
  wxLT $M_1 = (Q_1, \Sigma, \Delta, R_1, q_{1_0})$
  wLNT $M_2 = (Q_2, \Delta, \Gamma, R_2, q_{2_0})$
outputs
  wxLT $M_3 = ((Q_1 \times Q_2), \Sigma, \Gamma, R_3, (q_{1_0}, q_{2_0}))$ such that $M_3 = (\tau_{M_1}; \tau_{M_2})$.
complexity
  $O(|R_1| \max(|R_2|^{size(\tilde{u})}, |Q_2|))$, where $\tilde{u}$ is the largest right side tree in any rule in R_1

Let R_3 be of the form (R'_3, π)
$R_3 \leftarrow (\emptyset, \emptyset)$
$\Xi \leftarrow \{(q_{1_0}, q_{2_0})\}$ {seen states}
$\Psi \leftarrow \{(q_{1_0}, q_{2_0})\}$ {pending states}
while $\Psi \neq \emptyset$ **do**
  $(q_1, q_2) \leftarrow$any element of Ψ
  $\Psi \leftarrow \Psi \setminus \{(q_1, q_2)\}$
  for all $(q_1.y \xrightarrow{w_1} u) \in R_1$ **do**
    for all $(z, w_2) \in \text{COVER}(u, M_2, q_2)$ **do**
      for all $(q, x) \in yd(z) \cap ((Q_1 \times Q_2) \times X)$ **do**
        if $q \notin \Xi$ **then**
          $\Xi \leftarrow \Xi \cup \{q\}$
          $\Psi \leftarrow \Psi \cup \{q\}$
        $r \leftarrow ((q_1, q_2).y \rightarrow z)$
        $R'_3 \leftarrow R'_3 \cup \{r\}$
        $\pi(r) \leftarrow \pi(r) + (w_1 \cdot w_2)$
return M_3

4 Application of tree transducer cascades

What about the case of an input WRTG and a cascade of tree transducers? We will revisit the three strategies for accomplishing application discussed above for the string case.

In order for offline composition to be a viable strategy, the transducers in the cascade must be closed under composition. Unfortunately, of the classes that preserve recognizability, only wLNT is closed under composition (Gécseg and Steinby, 1984; Baker, 1979; Maletti et al., 2009; Fülöp and Vogler, 2009).

However, the general lack of composability of tree transducers does *not* preclude us from conducting forward application of a cascade. We revisit the bucket brigade approach, which in Section 2 appeared to be little more than a choice of composition order. As discussed previously, application of a single transducer involves an embedding, a composition, and a projection. The embedded WRTG is in the class wLNT, and the projection forms another WRTG. As long as every transducer in the cascade can be composed with a wLNT to its left or right, depending on the application type, application of a cascade is possible. Note that this embed-compose-project process is somewhat more burdensome than in the string case. For strings, application is obtained by a *single* embedding, a series of compositions, and a *single* projection, whereas application for trees is obtained by a series of (embed, compose, project) operations.

Algorithm 2 COVER

inputs
  $u \in T_\Delta(Q_1 \times X)$
  wT $M_2 = (Q_2, \Delta, \Gamma, R_2, q_{2_0})$
  state $q_2 \in Q_2$
outputs
  set of pairs (z, w) with $z \in T_\Gamma((Q_1 \times Q_2) \times X)$ formed by one or more successful runs on u by rules in R_2, starting from q_2, and $w \in \mathbb{R}^\infty_+$ the sum of the weights of all such runs.
complexity
  $O(|R_2|^{size(u)})$

if $u(\varepsilon)$ is of the form $(q_1, x) \in Q_1 \times X$ **then**
  $z_{init} \leftarrow ((q_1, q_2), x)$
else
  $z_{init} \leftarrow \perp$
$\Pi_{last} \leftarrow \{(z_{init}, \{((\varepsilon, \varepsilon), q_2)\}, 1)\}$
for all $v \in pos(u)$ such that $u(v) \in \Delta^{(k)}$ for some $k \geq 0$ in prefix order **do**
  $\Pi_v \leftarrow \emptyset$
  for all $(z, \theta, w) \in \Pi_{last}$ **do**
    for all $v' \in lv(z)$ such that $z(v') = \perp$ **do**
      for all $(\theta(v, v').u(v)(x_1, \ldots, x_k) \xrightarrow{w'} h) \in R_2$ **do**
        $\theta' \leftarrow \theta$
        Form substitution mapping $\varphi : (Q_2 \times X) \rightarrow T_\Gamma((Q_1 \times Q_2 \times X) \cup \{\perp\})$.
        for $i = 1$ **to** k **do**
          for all $v'' \in pos(h)$ such that $h(v'') = (q'_2, x_i)$ for some $q'_2 \in Q_2$ **do**
            $\theta'(vi, v'v'') \leftarrow q'_2$
            if $u(vi)$ is of the form $(q_1, x) \in Q_1 \times X$ **then**
              $\varphi(q'_2, x_i) \leftarrow ((q_1, q'_2), x)$
            else
              $\varphi(q'_2, x_i) \leftarrow \perp$
        $\Pi_v \leftarrow \Pi_v \cup \{(z[\overline{\varphi}(h)]_{v'}, \theta', w \cdot w')\}$
  $\Pi_{last} \leftarrow \Pi_v$
$Z \leftarrow \{z \mid (z, \theta, w) \in \Pi_{last}\}$
return $\{(z, \sum_{(z,\theta,w) \in \Pi_{last}} w) \mid z \in Z\}$

4.1 On-the-fly algorithms

We next consider on-the-fly algorithms for application. Similar to the string case, an on-the-fly approach is driven by a calling algorithm that periodically needs to know the productions in a WRTG with a common left side nonterminal. The embed-compose-project approach produces an entire application WRTG before any inference algorithm is run. In order to admit an on-the-fly approach we describe algorithms that only generate those productions in a WRTG that have a given left nonterminal. In this section we extend Definition 3.1 as follows: a WRTG is a 6-tuple $G = (N, \Sigma, P, n_0, \overline{M}, \overline{G})$ where N, Σ, P, and n_0 are defined as in Definition 3.1, and either $\overline{M} = \overline{G} = \emptyset$,[10] or $\overline{M}$ is a wxLNT and $\overline{G}$ is a normal form, chain production-free WRTG such that

[10] In which case the definition is functionally unchanged from before.

type	preserved?	source
w[x]T	No	See w[x]NT
w[x]LT	OQ	(Maletti, 2009)
w[x]NT	No	(Gécseg and Steinby, 1984)
wxLNT	Yes	(Fülöp et al., 2010)
wLNT	Yes	(Kuich, 1999)

(a) Preservation of forward recognizability

type	preserved?	source
w[x]T	No	See w[x]NT
w[x]LT	Yes	(Fülöp et al., 2010)
w[x]NT	No	(Maletti, 2009)
w[x]LNT	Yes	See w[x]LT

(b) Preservation of backward recognizability

Table 1: Preservation of forward and backward recognizability for various classes of top-down tree transducers. Here and elsewhere, the following abbreviations apply: w = weighted, x = extended LHS, L = linear, N = nondeleting, OQ = open question. Square brackets include a superposition of classes. For example, w[x]T signifies both wxT and wT.

Algorithm 3 PRODUCE

inputs
  WRTG $G_{in} = (N_{in}, \Delta, P_{in}, n_0, \overline{M}, \overline{G})$ such that $\overline{M} = (Q, \Sigma, \Delta, R, q_0)$ is a wxLNT and $\overline{G} = (N, \Sigma, P, n_0', M', G')$ is a WRTG in normal form with no chain productions
  $n_{in} \in N_{in}$
outputs
  WRTG G_{out} = $(N_{out}, \Delta, P_{out}, n_0, \overline{M}, \overline{G})$, such that $G_{in} \preceq G_{out}$ and $(n_{in} \xrightarrow{w} u) \in P_{out} \Leftrightarrow (n_{in} \xrightarrow{w} u) \in \overline{M}(\overline{G})^{\triangleright}$
complexity
  $O(|R||P|^{size(\tilde{y})})$, where $\tilde{y}$ is the largest left side tree in any rule in R

if P_{in} contains productions of the form $n_{in} \xrightarrow{w} u$ **then**
  return G_{in}
$N_{out} \leftarrow N_{in}$
$P_{out} \leftarrow P_{in}$
Let n_{in} be of the form (n, q), where $n \in N$ and $q \in Q$.
for all $(q.y \xrightarrow{w_1} u) \in R$ **do**
  for all $(\theta, w_2) \in$ REPLACE$(y, \overline{G}, n)$ **do**
    Form substitution mapping $\varphi : Q \times X \to T_{\Delta}(N \times Q)$ such that, for all $v \in yd(y)$ and $q' \in Q$, if there exist $n' \in N$ and $x \in X$ such that $\theta(v) = n'$ and $y(v) = x$, then $\varphi(q', x) = (n', q')$.
    $p' \leftarrow ((n, q) \xrightarrow{w_1 \cdot w_2} \overline{\varphi}(u))$
    for all $p \in$ NORM(p', N_{out}) **do**
      Let p be of the form $n_0 \xrightarrow{w} \delta(n_1, \ldots, n_k)$ for $\delta \in \Delta^{(k)}$.
      $N_{out} \leftarrow N_{out} \cup \{n_0, \ldots, n_k\}$
      $P_{out} \leftarrow P_{out} \cup \{p\}$
return CHAIN-REM(G_{out})

$G \preceq \overline{M}(\overline{G})^{\triangleright}$. In the latter case, G is a stand-in for $\overline{M}(\overline{G})^{\triangleright}$, analogous to the stand-ins for WSAs and WSTs described in Section 2.

Algorithm 3, PRODUCE, takes as input a WRTG $G_{in} = (N_{in}, \Delta, P_{in}, n_0, \overline{M}, \overline{G})$ and a desired nonterminal n_{in} and returns another WRTG, G_{out} that is different from G_{in} in that it has more productions, specifically those beginning with n_{in} that are in $\overline{M}(\overline{G})^{\triangleright}$. Algorithms using stand-ins should call PRODUCE to ensure the stand-in they are using has the desired productions beginning with the specific nonterminal. Note, then, that PRODUCE obtains the effect of forward applica-

Algorithm 4 REPLACE

inputs
  $y \in T_{\Sigma}(X)$
  WRTG $G = (N, \Sigma, P, n_0, \overline{M}, \overline{G})$ in normal form, with no chain productions
  $n \in N$
outputs
  set Π of pairs (θ, w) where θ is a mapping $pos(y) \to N$ and $w \in \mathbb{R}_+^{\infty}$, each pair indicating a successful run on y by productions in G, starting from n, and w is the weight of the run.
complexity
  $O(|P|^{size(y)})$

$\Pi_{last} \leftarrow \{(\{(\varepsilon, n)\}, 1)\}$
for all $v \in pos(y)$ such that $y(v) \notin X$ in prefix order **do**
  $\Pi_v \leftarrow \emptyset$
  for all $(\theta, w) \in \Pi_{last}$ **do**
    if $\overline{M} \neq \emptyset$ **and** $\overline{G} \neq \emptyset$ **then**
      $G \leftarrow$ PRODUCE$(G, \theta(v))$
    for all $(\theta(v) \xrightarrow{w'} y(v)(n_1, \ldots, n_k)) \in P$ **do**
      $\Pi_v \leftarrow \Pi_v \cup \{(\theta \cup \{(vi, n_i), 1 \leq i \leq k\}, w \cdot w')\}$
  $\Pi_{last} \leftarrow \Pi_v$
return Π_{last}

Algorithm 5 MAKE-EXPLICIT

inputs
  WRTG $G = (N, \Sigma, P, n_0, \overline{M}, \overline{G})$ in normal form
outputs
  WRTG $G' = (N', \Sigma, P', n_0, \overline{M}, \overline{G})$, in normal form, such that if $\overline{M} \neq \emptyset$ and $\overline{G} \neq \emptyset$, $L_{G'} = L_{\overline{M}(\overline{G})^{\triangleright}}$, and otherwise $G' = G$.
complexity
  $O(|P'|)$

$G' \leftarrow G$
$\Xi \leftarrow \{n_0\}$ {seen nonterminals}
$\Psi \leftarrow \{n_0\}$ {pending nonterminals}
while $\Psi \neq \emptyset$ **do**
  $n \leftarrow$any element of Ψ
  $\Psi \leftarrow \Psi \setminus \{n\}$
  if $\overline{M} \neq \emptyset$ **and** $\overline{G} \neq \emptyset$ **then**
    $G' \leftarrow$ PRODUCE(G', n)
  for all $(n \xrightarrow{w} \sigma(n_1, \ldots, n_k)) \in P'$ **do**
    for $i = 1$ to k **do**
      if $n_i \notin \Xi$ **then**
        $\Xi \leftarrow \Xi \cup \{n_i\}$
        $\Psi \leftarrow \Psi \cup \{n_i\}$
return G'

g_0
$g_0 \xrightarrow{w_1} \sigma(g_0, g_1)$
$g_0 \xrightarrow{w_2} \alpha \quad g_1 \xrightarrow{w_3} \alpha$

(a) Input WRTG G

a_0
$a_0.\sigma(x_1, x_2) \xrightarrow{w_4} \sigma(a_0.x_1, a_1.x_2)$
$a_0.\sigma(x_1, x_2) \xrightarrow{w_5} \psi(a_2.x_1, a_1.x_2)$
$a_0.\alpha \xrightarrow{w_6} \alpha \quad a_1.\alpha \xrightarrow{w_7} \alpha \quad a_2.\alpha \xrightarrow{w_8} \rho$

(b) First transducer M_A in the cascade

b_0
$b_0.\sigma(x_1, x_2) \xrightarrow{w_9} \sigma(b_0.x_1, b_0.x_2)$
$b_0.\alpha \xrightarrow{w_{10}} \alpha$

(c) Second transducer M_B in the cascade

$g_0a_0 \xrightarrow{w_1 \cdot w_4} \sigma(g_0a_0, g_1a_1)$
$g_0a_0 \xrightarrow{w_1 \cdot w_5} \psi(g_0a_2, g_1a_1)$
$g_0a_0 \xrightarrow{w_2 \cdot w_6} \alpha \quad g_1a_1 \xrightarrow{w_3 \cdot w_7} \alpha$

(d) Productions of $M_A(G)^{\triangleright}$ built as a consequence of building the complete $M_B(M_A(G)^{\triangleright})^{\triangleright}$

$g_0a_0b_0$
$g_0a_0b_0 \xrightarrow{w_1 \cdot w_4 \cdot w_9} \sigma(g_0a_0b_0, g_1a_1b_0)$
$g_0a_0b_0 \xrightarrow{w_2 \cdot w_6 \cdot w_{10}} \alpha \quad g_1a_1b_0 \xrightarrow{w_3 \cdot w_7 \cdot w_{10}} \alpha$

(e) Complete $M_B(M_A(G)^{\triangleright})^{\triangleright}$

Figure 2: Forward application through a cascade of tree transducers using an on-the-fly method.

tion in an on-the-fly manner.[11] It makes calls to REPLACE, which is presented in Algorithm 4, as well as to a NORM algorithm that ensures normal form by replacing a single production not in normal form with several normal-form productions that can be combined together (Alexandrakis and Bozapalidis, 1987) and a CHAIN-REM algorithm that replaces a WRTG containing chain productions with an equivalent WRTG that does not (Mohri, 2009).

As an example of stand-in construction, consider the invocation PRODUCE(G_1, g_0a_0), where G_1 = ($\{g_0a_0\}$, $\{\sigma, \psi, \alpha, \rho\}$, $\emptyset$, g_0a_0, M_A, G), G is in Figure 2a,[12] and M_A is in 2b. The stand-in WRTG that is output contains the first three of the four productions in Figure 2d.

To demonstrate the use of on-the-fly application in a cascade, we next show the effect of PRODUCE when used with the cascade $G \circ M_A \circ M_B$, where M_B is in Figure 2c. Our driving algorithm in this case is Algorithm 5, MAKE-

$r_{\text{JJ}}.\text{JJ}(x_1, x_2, x_3) \rightarrow \text{JJ}(r_{\text{DT}}.x_1, r_{\text{JJ}}.x_2, r_{\text{VB}}.x_3)$
$r_{\text{VB}}.\text{VB}(x_1, x_2, x_3) \rightarrow \text{VB}(r_{\text{NNPS}}.x_1, r_{\text{NN}}.x_3, r_{\text{VB}}.x_2)$
t."gentle" $\rightarrow$ "gentle"

(a) Rotation rules

$i_{\text{VB}}.\text{NN}(x_1, x_2) \rightarrow \text{NN}(\text{INS } i_{\text{NN}}.x_1, i_{\text{NN}}.x_2)$
$i_{\text{VB}}.\text{NN}(x_1, x_2) \rightarrow \text{NN}(i_{\text{NN}}.x_1, i_{\text{NN}}.x_2)$
$i_{\text{VB}}.\text{NN}(x_1, x_2) \rightarrow \text{NN}(i_{\text{NN}}.x_1, i_{\text{NN}}.x_2, \text{INS})$

(b) Insertion rules

$t.\text{VB}(x_1, x_2, x_3) \rightarrow \text{X}(t.x_1, t.x_2, t.x_3)$
t."gentleman" $\rightarrow$ j1
t."gentleman" $\rightarrow$ EPS
t.INS $\rightarrow$ j1
t.INS $\rightarrow$ j2

(c) Translation rules

Figure 3: Example rules from transducers used in decoding experiment. j1 and j2 are Japanese words.

EXPLICIT, which simply generates the full application WRTG using calls to PRODUCE. The input to MAKE-EXPLICIT is G_2 = ($\{g_0a_0b_0\}$, $\{\sigma, \alpha\}$, $\emptyset$, $g_0a_0b_0$, M_B, G_1).[13] MAKE-EXPLICIT calls PRODUCE(G_2, $g_0a_0b_0$). PRODUCE then seeks to cover $b_0.\sigma(x_1, x_2) \xrightarrow{w_9} \sigma(b_0.x_1, b_0.x_2)$ with productions from G_1, which is a stand-in for $M_A(G)^{\triangleright}$. At line 14 of REPLACE, G_1 is improved so that it has the appropriate productions. The productions of $M_A(G)^{\triangleright}$ that must be built to form the complete $M_B(M_A(G)^{\triangleright})^{\triangleright}$ are shown in Figure 2d. The complete $M_B(M_A(G)^{\triangleright})^{\triangleright}$ is shown in Figure 2e. Note that because we used this on-the-fly approach, we were able to avoid building all the productions in $M_A(G)^{\triangleright}$; in particular we did not build $g_0a_2 \xrightarrow{w_2 \cdot w_8} \rho$, while a bucket brigade approach would have built this production. We have also designed an analogous on-the-fly PRODUCE algorithm for backward application on linear WTT.

We have now defined several on-the-fly and bucket brigade algorithms, and also discussed the possibility of embed-compose-project and offline composition strategies to application of cascades of tree transducers. Tables 2a and 2b summarize the available methods of forward and backward application of cascades for recognizability-preserving tree transducer classes.

5 Decoding Experiments

The main purpose of this paper has been to present novel algorithms for performing application. However, it is important to demonstrate these algorithms on real data. We thus demonstrate bucket-brigade and on-the-fly backward application on a typical NLP task cast as a cascade of wLNT. We adapt the Japanese-to-English transla-

[11] Note further that it allows forward application of class wxLNT, something the embed-compose-project approach did not allow.

[12] By convention the initial nonterminal and state are listed first in graphical depictions of WRTGs and WXTTs.

[13] Note that G_2 is the initial stand-in for $M_B(M_A(G)^{\triangleright})^{\triangleright}$, since G_1 is the initial stand-in for $M_A(G)^{\triangleright}$.

method	WST	wxLNT	wLNT
oc	√	×	√
bb	√	×	√
otf	√	√	√

(a) Forward application

method	WST	wxLT	wLT	wxLNT	wLNT
oc	√	×	×	×	√
bb	√	√	√	√	√
otf	√	√	√	√	√

(b) Backward application

Table 2: Transducer types and available methods of forward and backward application of a cascade. oc = offline composition, bb = bucket brigade, otf = on the fly.

tion model of Yamada and Knight (2001) by transforming it from an English-tree-to-Japanese-string model to an English-tree-to-Japanese-*tree* model. The Japanese trees are unlabeled, meaning they have syntactic structure but all nodes are labeled "X". We then cast this modified model as a cascade of LNT tree transducers. Space does not permit a detailed description, but some example rules are in Figure 3. The rotation transducer $\mathcal{R}$, a sample of which is in Figure 3a, has 6,453 rules, the insertion transducer $\mathcal{I}$, Figure 3b, has 8,122 rules, and the translation transducer, $\mathcal{T}$, Figure 3c, has 37,311 rules.

We add an English syntax language model $\mathcal{L}$ to the cascade of transducers just described to better simulate an actual machine translation decoding task. The language model is cast as an identity WTT and thus fits naturally into the experimental framework. In our experiments we try several different language models to demonstrate varying performance of the application algorithms. The most realistic language model is a PCFG. Each rule captures the probability of a particular sequence of child labels given a parent label. This model has 7,765 rules.

To demonstrate more extreme cases of the usefulness of the on-the-fly approach, we build a language model that recognizes exactly the 2,087 trees in the training corpus, each with equal weight. It has 39,455 rules. Finally, to be ultra-specific, we include a form of the "specific" language model just described, but only allow the English counterpart of the particular Japanese sentence being decoded in the language.

The goal in our experiments is to apply a single tree t backward through the cascade $\mathcal{L} \circ \mathcal{R} \circ \mathcal{I} \circ \mathcal{T} \circ t$ and find the 1-best path in the application WRTG. We evaluate the speed of each approach: bucket brigade and on-the-fly. The algorithm we use to obtain the 1-best path is a modification of the k-best algorithm of Pauls and Klein (2009). Our algorithm finds the 1-best path in a WRTG and admits an on-the-fly approach.

The results of the experiments are shown in Table 3. As can be seen, on-the-fly application is generally faster than the bucket brigade, about double the speed per sentence in the traditional

LM type	method	time/sentence
pcfg	bucket	28s
pcfg	otf	17s
exact	bucket	>1m
exact	otf	24s
1-sent	bucket	2.5s
1-sent	otf	.06s

Table 3: Timing results to obtain 1-best from application through a weighted tree transducer cascade, using on-the-fly vs. bucket brigade backward application techniques. pcfg = model recognizes any tree licensed by a pcfg built from observed data, exact = model recognizes each of 2,000+ trees with equal weight, 1-sent = model recognizes exactly one tree.

experiment that uses an English PCFG language model. The results for the other two language models demonstrate more keenly the potential advantage that an on-the-fly approach provides—the simultaneous incorporation of information from all models allows application to be done more effectively than if each information source is considered in sequence. In the "exact" case, where a very large language model that simply recognizes each of the 2,087 trees in the training corpus is used, the final application is so large that it overwhelms the resources of a 4gb MacBook Pro, while the on-the-fly approach does not suffer from this problem. The "1-sent" case is presented to demonstrate the ripple effect caused by using on-the fly. In the other two cases, a very large language model generally overwhelms the timing statistics, regardless of the method being used. But a language model that represents exactly one sentence is very small, and thus the effects of simultaneous inference are readily apparent—the time to retrieve the 1-best sentence is reduced by two orders of magnitude in this experiment.

6 Conclusion

We have presented algorithms for forward and backward application of weighted tree transducer cascades, including on-the-fly variants, and demonstrated the benefit of an on-the-fly approach to application. We note that a more formal approach to application of WTTs is being developed,

independent from these efforts, by Fülöp et al. (2010).

Acknowledgments

We are grateful for extensive discussions with Andreas Maletti. We also appreciate the insights and advice of David Chiang, Steve DeNeefe, and others at ISI in the preparation of this work. Jonathan May and Kevin Knight were supported by NSF grants IIS-0428020 and IIS-0904684. Heiko Vogler was supported by DFG VO 1011/5-1.

References

Athanasios Alexandrakis and Symeon Bozapalidis. 1987. Weighted grammars and Kleene's theorem. *Information Processing Letters*, 24(1):1–4.

Brenda S. Baker. 1979. Composition of top-down and bottom-up tree transductions. *Information and Control*, 41(2):186–213.

Zoltán Ésik and Werner Kuich. 2003. Formal tree series. *Journal of Automata, Languages and Combinatorics*, 8(2):219–285.

Zoltán Fülöp and Heiko Vogler. 2009. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 9, pages 313–404. Springer-Verlag.

Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. 2010. Backward and forward application of weighted extended tree transducers. Unpublished manuscript.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

Liang Huang and David Chiang. 2005. Better k-best parsing. In Harry Bunt, Robert Malouf, and Alon Lavie, editors, *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 53–64, Vancouver, October. Association for Computational Linguistics.

Werner Kuich. 1998. Formal power series over trees. In Symeon Bozapalidis, editor, *Proceedings of the 3rd International Conference on Developments in Language Theory (DLT)*, pages 61–101, Thessaloniki, Greece. Aristotle University of Thessaloniki.

Werner Kuich. 1999. Tree transducers and formal tree series. *Acta Cybernetica*, 14:135–149.

Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. 2009. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430.

Andreas Maletti. 2006. Compositions of tree series transformations. *Theoretical Computer Science*, 366:248–271.

Andreas Maletti. 2008. Compositions of extended top-down tree transducers. *Information and Computation*, 206(9–10):1187–1196.

Andreas Maletti. 2009. Personal Communication.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–312.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 6, pages 213–254. Springer-Verlag.

Adam Pauls and Dan Klein. 2009. K-best A* parsing. In Keh-Yih Su, Jian Su, Janyce Wiebe, and Haizhou Li, editors, *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 958–966, Suntec, Singapore, August. Association for Computational Linguistics.

Fernando Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 15, pages 431–453. MIT Press, Cambridge, MA.

William A. Woods. 1980. Cascaded ATN grammars. *American Journal of Computational Linguistics*, 6(1):1–12.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, July. Association for Computational Linguistics.

A Tree Transducer Model for Synchronous Tree-Adjoining Grammars

Andreas Maletti
Universitat Rovira i Virgili
Avinguda de Catalunya 25, 43002 Tarragona, Spain.
andreas.maletti@urv.cat

Abstract

A characterization of the expressive power of synchronous tree-adjoining grammars (STAGs) in terms of tree transducers (or equivalently, synchronous tree substitution grammars) is developed. Essentially, a STAG corresponds to an extended tree transducer that uses explicit substitution in both the input and output. This characterization allows the easy integration of STAG into toolkits for extended tree transducers. Moreover, the applicability of the characterization to several representational and algorithmic problems is demonstrated.

1 Introduction

Machine translation has seen a multitude of formal translation models. Here we focus on syntax-based (or tree-based) models. One of the oldest models is the *synchronous context-free grammar* (Aho and Ullman, 1972). It is clearly too weak as a syntax-based model, but found use in the string-based setting. *Top-down tree transducers* (Rounds, 1970; Thatcher, 1970) have been heavily investigated in the formal language community (Gécseg and Steinby, 1984; Gécseg and Steinby, 1997), but as argued by Shieber (2004) they are still too weak for syntax-based machine translation. Instead Shieber (2004) proposes *synchronous tree substitution grammars* (STSGs) and develops an equivalent bimorphism (Arnold and Dauchet, 1982) characterization. This characterization eventually led to the rediscovery of *extended tree transducers* (Graehl and Knight, 2004; Knight and Graehl, 2005; Graehl et al., 2008), which are essentially as powerful as STSG. They had been studied already by Arnold and Dauchet (1982) in the form of bimorphisms, but received little attention until rediscovered.

Shieber (2007) claims that even STSGs might be too simple to capture naturally occuring translation phenomena. Instead Shieber (2007) suggests a yet more powerful mechanism, *synchronous tree-adjoining grammars* (STAGs) as introduced by Shieber and Schabes (1990), that can capture certain (mildly) context-sensitive features of natural language. In the tradition of Shieber (2004), a characterization of the power of STAGs in terms of bimorphims was developed by Shieber (2006). The bimorphisms used are rather unconventional because they consist of a regular tree language and two embedded tree transducers (instead of two tree homomorphisms). Such embedded tree transducers (Shieber, 2006) are particular macro tree transducers (Courcelle and Franchi-Zannettacci, 1982; Engelfriet and Vogler, 1985).

In this contribution, we try to unify the picture even further. We will develop a tree transducer model that can simulate STAGs. It turns out that the adjunction operation of an STAG can be explained easily by explicit substitution. In this sense, the slogan that an STAG is an STSG with adjunction, which refers to the syntax, also translates to the semantics. We prove that any tree transformation computed by an STAG can also be computed by an STSG using explicit substitution. Thus, a simple evaluation procedure that performs the explicit substitution is all that is needed to simulate an STAG in a toolkit for STSGs or extended tree transducers like TIBURON by May and Knight (2006).

We show that some standard algorithms on STAG can actually be run on the constructed STSG, which often is simpler and better understood. Further, it might be easier to develop new algorithms with the alternative characterization, which we demonstrate with a product construction for input restriction in the spirit of Nederhof (2009). Finally, we also present a complete tree transducer model that is as powerful as STAG, which is an extension of the embedded tree transducers of Shieber (2006).

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1067–1076,
Uppsala, Sweden, 11-16 July 2010.

2 Notation

We quickly recall some central notions about trees, tree languages, and tree transformations. For a more in-depth discussion we refer to Gécseg and Steinby (1984) and Gécseg and Steinby (1997). A finite set Σ of labels is an alphabet. The set of all strings over that alphabet is Σ^* where ε denotes the empty string. To simplify the presentation, we assume an infinite set $X = \{x_1, x_2, \dots\}$ of variables. Those variables are syntactic and represent only themselves. In particular, they are all different. For each $k \geq 0$, we let $X_k = \{x_1, \dots, x_k\}$. We can also form trees over the alphabet Σ. To allow some more flexibility, we will also allow leaves from a special set V. Formally, a Σ-tree over V is either:

- a leaf labeled with an element of $v \in \Sigma \cup V$, or
- a node that is labeled with an element of Σ with $k \geq 1$ children such that each child is a Σ-tree over V itself.[1]

The set of all Σ-trees over V is denoted by $T_\Sigma(V)$. We just write T_Σ for $T_\Sigma(\emptyset)$. The trees in Figure 1 are, for example, elements of $T_\Delta(Y)$ where

$$\Delta = \{\text{S}, \text{NP}, \text{VP}, \text{V}, \text{DT}, \text{N}\}$$
$$Y = \{\text{saw}, \text{the}\} \ .$$

We often present trees as terms. A leaf labeled v is simply written as v. The tree with a root node labeled σ is written $\sigma(t_1, \dots, t_k)$ where $t_1, \dots, t_k$ are the term representations of its k children.

A tree language is any subset of $T_\Sigma(V)$ for some alphabet Σ and set V. Given another alphabet Δ and a set Y, a tree transformation is a relation $\tau \subseteq T_\Sigma(V) \times T_\Delta(Y)$. In many of our examples we have $V = \emptyset = Y$. Occasionally, we also speak about the translation of a tree transformation $\tau \subseteq T_\Sigma \times T_\Delta$. The *translation* of τ is the relation $\{(\mathrm{yd}(t), \mathrm{yd}(u)) \mid (t, u) \in \tau\}$ where $\mathrm{yd}(t)$, the *yield* of t, is the sequence of leaf labels in a left-to-right tree traversal of t. The yield of the third tree in Figure 1 is "the N saw the N". Note that the translation is a relation $\tau' \subseteq \Sigma^* \times \Delta^*$.

3 Substitution

A standard operation on (labeled) trees is *substitution*, which replaces leaves with a specified label in one tree by another tree. We write $t[u]_A$ for (the result of) the substitution that replaces all leaves labeled A in the tree t by the tree u. If $t \in T_\Sigma(V)$ and $u \in T_\Delta(Y)$, then $t[u]_A \in T_{\Sigma\cup\Delta}(V \cup Y)$. We often use the variables of $X = \{x_1, x_2, \dots\}$ as substitution points and write $t[u_1, \dots, u_k]$ instead of $(\cdots(t[u_1]_{x_1})\dots)[u_k]_{x_k}$.

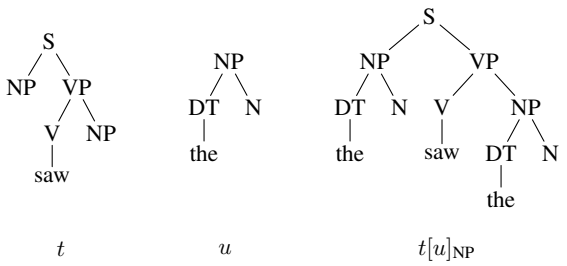


Figure 1: A substitution.

An example substitution is shown in Figure 1. The figure also illustrates a common problem with substitution. Occasionally, it is not desirable to replace all leaves with a certain label by the same tree. In the depicted example, we might want to replace one 'NP' by a different tree, which cannot be achieved with substitution. Clearly, this problem is avoided if the source tree t contains only one leaf labeled A. We call a tree *A-proper* if it contains exactly one leaf with label A.[2] The subset $C_\Sigma(X_k) \subseteq T_\Sigma(X_k)$ contains exactly those trees of $T_\Sigma(X_k)$ that are x_i-proper for every $1 \leq i \leq k$. For example, the tree t of Figure 1 is 'saw'-proper, and the tree u of Figure 1 is 'the'- and 'N'-proper.

In this contribution, we will also use substitution as an explicit operator. The tree $t[u]_{\text{NP}}$ in Figure 1 only shows the result of the substitution. It cannot be infered from the tree alone, how it was obtained (if we do not know t and u).[3] To make substitution explicit, we use the special binary symbols $\cdot[\cdot]_A$ where A is a label. Those symbols will always be used with exactly two children (i.e., as binary symbols). Since this property can easily be checked by all considered devices, we ignore trees that use those symbols in a non-binary manner. For every set Σ of labels, we let $\overline{\Sigma} = \Sigma \cup \{\cdot[\cdot]_A \mid A \in \Sigma\}$ be the extended set of labels containing also the substition symbols. The substitution of Figure 1 can then be ex-

[1] Note that we do not require the symbols to have a fixed rank; i.e., a symbol does not determine its number of children.

[2] A-proper trees are sometimes also called A-context in the literature.

[3] This remains true even if we know that the participating trees t and u are A-proper and the substitution $t[u]_A$ replacing leaves labeled A was used. This is due to the fact that, in general, the root label of u need not coincide with A.

pressed as the tree $\cdot[\cdot]_{\text{NP}}(t, u)$. To obtain $t[u]_{\text{NP}}$ (the right-most tree in Figure 1), we have to evaluate $\cdot[\cdot]_{\text{NP}}(t, u)$. However, we want to replace only one leaf at a time. Consequently, we restrict the evaluation of $\cdot[\cdot]_A(t, u)$ such that it applies only to trees t whose evaluation is A-proper. To enforce this restriction, we introduce an error signal $\perp$, which we assume not to occur in any set of labels. Let Σ be the set of labels. Then we define the function $\cdot^{\text{E}} \colon T_{\overline{\Sigma}} \to T_\Sigma \cup \{\perp\}$ by[4]

$$\sigma(t_1, \ldots, t_k)^{\text{E}} = \sigma(t_1^{\text{E}}, \ldots, t_k^{\text{E}})$$

$$\cdot[\cdot]_A(t, u)^{\text{E}} = \begin{cases} t^{\text{E}}[u^{\text{E}}]_A & \text{if } t^{\text{E}} \text{ is } A\text{-proper} \\ \perp & \text{otherwise} \end{cases}$$

for every $k \geq 0$, $\sigma \in \Sigma$, and $t, t_1, \ldots, t_k, u \in T_{\overline{\Sigma}}$.[5] We generally discard all trees that contain the error signal $\perp$. Since the devices that we will study later can also check the required A-properness using their state behavior, we generally do not discuss trees with error symbols explicitly.

4 Extended tree transducer

An extended tree transducer is a theoretical model that computes a tree transformation. Such transducers have been studied first by Arnold and Dauchet (1982) in a purely theoretic setting, but were later applied in, for example, machine translation (Knight and Graehl, 2005; Knight, 2007; Graehl et al., 2008; Graehl et al., 2009). Their popularity in machine translation is due to Shieber (2004), in which it is shown that extended tree transducers are essentially (up to a relabeling) as expressive as synchronous tree substitution grammars (STSG). We refer to Chiang (2006) for an introduction to synchronous devices.

Let us recall the formal definition. An *extended tree transducer* (for short: XTT)[6] is a system $M = (Q, \Sigma, \Delta, I, R)$ where

- Q is a finite set of *states*,
- Σ and Δ are alphabets of *input* and *output symbols*, respectively,
- $I \subseteq Q$ is a set of *initial states*, and
- R is a finite set of *rules* of the form

$$(q, l) \to (q_1 \cdots q_k, r)$$

where $k \geq 0$, $l \in C_\Sigma(X_k)$, and $r \in C_\Delta(X_k)$.

Recall that any tree of $C_\Sigma(X_k)$ contains each variable of $X_k = \{x_1, \ldots, x_k\}$ exactly once. In graphical representations of a rule

$$(q, l) \to (q_1 \cdots q_k, r) \in R \ ,$$

we usually

- add the state q as root node of the left-hand side[7], and
- add the states $q_1, \ldots, q_k$ on top of the nodes labeled $x_1, \ldots, x_k$, respectively, in the right-hand side of the rule.

Some example rules are displayed in Figure 2.

The rules are applied in the expected way (as in a term-rewrite system). The only additional feature are the states of Q, which can be used to control the derivation. A sentential form is a tree that contains exclusively output symbols towards the root and remaining parts of the input headed by a state as leaves. A derivation step starting from ξ then consists in

- selecting a leaf of ξ with remaining input symbols,
- matching the state q and the left-hand side l of a rule $(q, l) \to (q_1 \cdots q_k, r) \in R$ to the state and input tree stored in the leaf, thus matching input subtrees $t_1, \ldots, t_k$ to the variables $x_1, \ldots, x_k$,
- replacing all the variables $x_1, \ldots, x_k$ in the right-hand side r by the matched input subtrees $q_1(t_1), \ldots, q_k(t_k)$ headed by the corresponding state, respectively, and
- replacing the selected leaf in ξ by the tree constructed in the previous item.

The process is illustrated in Figure 3.

Formally, a sentential form of the XTT M is a tree of $\text{SF} = T_\Delta(Q(T_\Sigma))$ where

$$Q(T_\Sigma) = \{q(t) \mid q \in Q, t \in T_\Sigma\} \ .$$

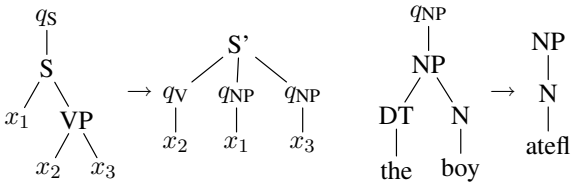


Figure 2: Example rules taken from Graehl et al. (2009). The term representation of the first rule is $(q_{\text{S}}, \text{S}(x_1, \text{VP}(x_2, x_3))) \to (w, \text{S}'(x_2, x_1, x_3))$ where $w = q_{\text{NP}} q_{\text{V}} q_{\text{NP}}$.

[4] Formally, we should introduce an evaluation function for each alphabet Σ, but we assume that the alphabet can be infered.

[5] This evaluation is a special case of a *yield*-mapping (Engelfriet and Vogler, 1985).

[6] Using the notions of Graehl et al. (2009) our extended tree transducers are linear, nondeleting extended top-down tree transducers.

[7] States are thus also special symbols that are exclusively used as unary symbols.

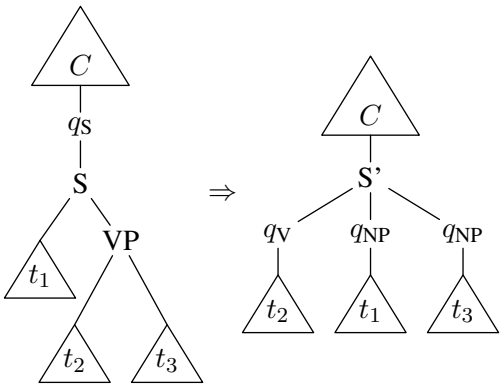


Figure 3: Illustration of a derivation step of an XTT using the left rule of Figure 2.

Given $\xi, \zeta \in \mathrm{SF}$, we write $\xi \Rightarrow \zeta$ if there exist $C \in C_\Delta(X_1)$, $t_1, \ldots, t_k \in T_\Sigma$, and a rule $(q, l) \to (q_1 \cdots q_k, r) \in R$ such that

- $\xi = C[q(l[t_1, \ldots, t_k])]$ and
- $\zeta = C[r[q_1(t_1), \ldots, q_k(t_k)]]$.

The tree transformation computed by M is the relation

$$\tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I \colon q(t) \Rightarrow^* u\}$$

where $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$. In other words, the tree t can be transformed into u if there exists an initial state q such that we can derive u from $q(t)$ in several derivation steps.

We refer to Arnold and Dauchet (1982), Graehl et al. (2008), and Graehl et al. (2009) for a more detailed exposition to XTT.

5 Synchronous tree-adjoining grammar

XTT are a simple, natural model for tree transformations, however they are not suitably expressive for all applications in machine translation (Shieber, 2007). In particular, all tree transformations of XTT have a certain locality condition, which yields that the input tree and its corresponding translation cannot be separated by an unbounded distance. To overcome this problem and certain dependency problems, Shieber and Schabes (1990) and Shieber (2007) suggest a stronger model called *synchronous tree-adjoining grammar* (STAG), which in addition to the substitution operation of STSG (Chiang, 2005) also has an adjoining operation.

Let us recall the model in some detail. A tree-adjoining grammar essentially is a regular tree grammar (Gécseg and Steinby, 1984; Gécseg and Steinby, 1997) enhanced with an adjunction operation. Roughly speaking, an adjunction replaces a node (not necessarily a leaf) by an *auxiliary tree*, which has exactly one distinguished *foot node*. The original children of the replaced node will become the children of the foot node after adjunction. Traditionally, the root label and the label of the foot node coincide in an auxiliary tree aside from a star index that marks the foot node. For example, if the root node of an auxiliary tree is labeled A, then the foot node is traditionally labeled $A_\star$. The star index is not reproduced once adjoined. Formally, the adjunction of the auxiliary tree u with root label A (and foot node label $A_\star$) into a tree $t = C[A(t_1, \ldots, t_k)]$ with $C \in C_\Sigma(X_1)$ and $t_1, \ldots, t_k \in T_\Sigma$ is

$$C[u[A(t_1, \ldots, t_k)]_{A_\star}] \ .$$

Adjunction is illustrated in Figure 4.

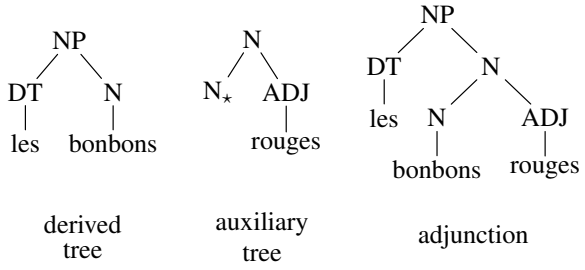


Figure 4: Illustration of an adjunction taken from Nesson et al. (2008).

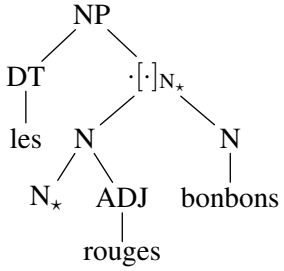


Figure 5: Illustration of the adjunction of Figure 4 using explicit substitution.

We note that adjunction can easily be expressed using explicit substitution. Essentially, only an additional node with the adjoined subtree is added. The result of the adjunction of Figure 4 using explicit substitution is displayed in Figure 5.

To simplify the development, we will make some assumptions on all tree-adjoining grammars (and synchronous tree-adjoining grammars). A *tree-adjoining grammar* (TAG) is a finite set of *initial trees* and a finite set of auxiliary trees. Our

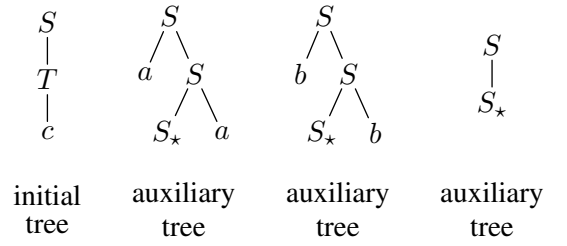


Figure 6: A TAG for the copy string language $\{wcw \mid w \in \{a, b\}^*\}$ taken from Shieber (2006).

TAG do not use substitution, but only adjunction. A derivation is a chain of trees that starts with an initial tree and each derived tree is obtained from the previous one in the chain by adjunction of an auxiliary tree. As in Shieber (2006) we assume that all adjunctions are mandatory; i.e., if an auxiliary tree can be adjoined, then we need to make an adjunction. Thus, a derivation starting from an initial tree to a derived tree is *complete* if no adjunction is possible in the derived tree. Moreover, we assume that to each node only one adjunction can be applied. This is easily achieved by labeling the root of each adjoined auxiliary tree by a special marker. Traditionally, the root label A of an auxiliary tree is replaced by $A_\emptyset$ once adjoined. Since we assume that there are no auxiliary trees with such a root label, no further adjunction is possible at such nodes. Another effect of this restriction is that the number of *operable nodes* (i.e., the nodes to which an adjunction must still be applied) is known at any given time.[8] A full TAG with our restrictions is shown in Figure 6.

Intuitively, a *synchronous tree-adjoining grammar* (STAG) is essentially a pair of TAGs. The synchronization is achieved by pairing the initial trees and the auxiliary trees. In addition, for each such pair (t, u) of trees, there exists a bijection between the operable nodes of t and u. Such nodes in bijection are *linked* and the links are preserved in derivations, in which we now use pairs of trees as sentential forms. In graphical representations we often indicate this bijection with integers; i.e., two nodes marked with the same integer are linked. A pair of auxiliary trees is then adjoined to linked nodes (one in each tree of the sentential form) in the expected manner. We will avoid a formal definition here, but rather present an example STAG and a derivation with it in Figures 7 and 8. For a

[8] Without the given restrictions, this number cannot be determined easily because no or several adjunctions can take place at a certain node.

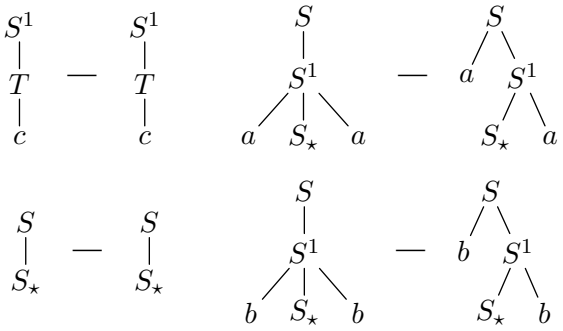


Figure 7: STAG that computes the translation $\{(wcw^{\mathrm{R}}, wcw) \mid w \in \{a, b\}^*\}$ where w^{R} is the reverse of w.

STAG G we write τ_G for the tree transformation computed by G.

6 Main result

In this section, we will present our main result. Essentially, it states that a STAG is as powerful as a STSG using explicit substitution. Thus, for every tree transformation computed by a STAG, there is an extended tree transducer that computes a representation of the tree transformation using explicit substitution. The converse is also true. For every extended tree transducer M that uses explicit substitution, we can construct a STAG that computes the tree transformation represented by τ_M up to a relabeling (a mapping that consistently replaces node labels throughout the tree). The additional relabeling is required because STAGs do not have states. If we replace the extended tree transducer by a STSG, then the result holds even without the relabeling.

Theorem 1 *For every STAG G, there exists an extended tree transducer M such that*

$$\tau_G = \{(t^{\mathrm{E}}, u^{\mathrm{E}}) \mid (t, u) \in \tau_M\} \ .$$

Conversely, for every extended tree transducer M, there exists a STAG G such that the above relation holds up to a relabeling.

6.1 Proof sketch

The following proof sketch is intended for readers that are familiar with the literature on embedded tree transducers, macro tree transducers, and bimorphisms. It can safely be skipped because we will illustrate the relevant construction on our example after the proof sketch, which contains the outline for the correctness.

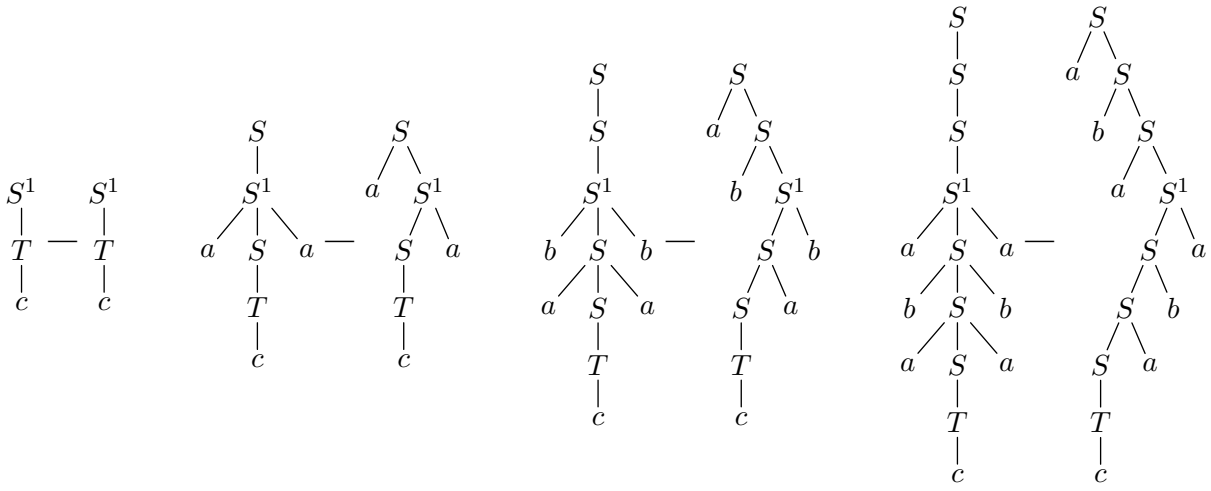


Figure 8: An incomplete derivation using the STAG of Figure 7.

Let $\tau \subseteq T_\Sigma \times T_\Delta$ be a tree transformation computed by a STAG. By Shieber (2006) there exists a regular tree language $L \subseteq T_\Gamma$ and two functions $e_1 \colon T_\Gamma \to T_\Sigma$ and $e_2 \colon T_\Gamma \to T_\Delta$ such that $\tau = \{(e_1(t), e_2(t)) \mid t \in L\}$. Moreover, e_1 and e_2 can be computed by *embedded tree transducers* (Shieber, 2006), which are particular 1-state, deterministic, total, 1-parameter, linear, and nondeleting macro tree transducers (Courcelle and Franchi-Zannettacci, 1982; Engelfriet and Vogler, 1985). In fact, the converse is also true up to a relabeling, which is also shown in Shieber (2006). The outer part of Figure 9 illustrates these relations. Finally, we remark that all involved constructions are effective.

Using a result of Engelfriet and Vogler (1985), each embedded tree transducer can be decomposed into a top-down tree transducer (Gécseg and Steinby, 1984; Gécseg and Steinby, 1997) and a yield-mapping. In our particular case, the top-down tree transducers are linear and nondeleting homomorphisms h_1 and h_2. Linearity and nondeletion are inherited from the corresponding properties of the macro tree transducer. The properties '1-state', 'deterministic', and 'total' of the macro tree transducer ensure that the obtained top-down tree transducer is also 1-state, deterministic, and total, which means that it is a homomorphism. Finally, the 1-parameter property yields that the used substitution symbols are binary (as our substitution symbols $\cdot[\cdot]_A$). Consequently, the yield-mapping actually coincides with our evaluation. Again, this decomposition actually is a characterization of embedded tree transducers. Now the set $\{(h_1(t), h_2(t)) \mid t \in L\}$ can be computed

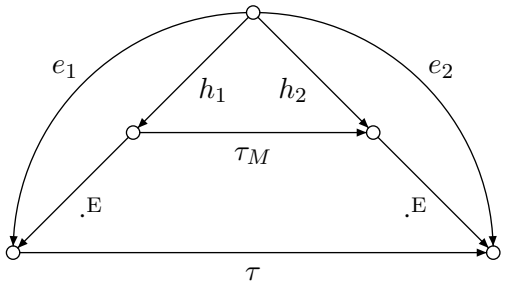


Figure 9: Illustration of the proof sketch.

by an extended tree transducer M due to results of Shieber (2004) and Maletti (2008). More precisely, every extended tree transducer computes such a set, so that also this step is a characterization. Thus we obtain that τ is an evaluation of a tree transformation computed by an extended tree transducer, and moreover, for each extended tree transducer, the evaluation can be computed (up to a relabeling) by a STAG. The overall proof structure is illustrated in Figure 9.

6.2 Example

Let us illustrate one direction (the construction of the extended tree transducer) on our example STAG of Figure 7. Essentially, we just prepare all operable nodes by inserting an explicit substitution just on top of them. The first subtree of that substitution will either be a variable (in the left-hand side of a rule) or a variable headed by a state (in the right-hand side of a rule). The numbers of the variables encode the links of the STAG. Two example rules obtained from the STAG of Figure 7 are presented in Figure 10. Using all XTT rules constructed for the STAG of Figure 7, we present

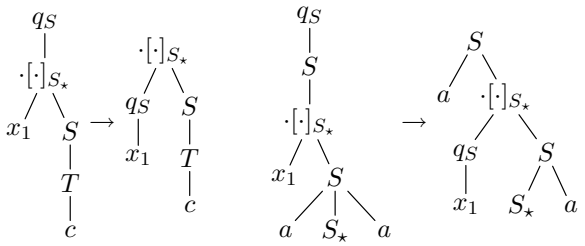


Figure 10: Two constructed XTT rules.

a complete derivation of the XTT in Figure 11 that (up to the final step) matches the derivation of the STAG in Figure 8. The matching is achieved by the evaluation $\cdot^{\mathrm{E}}$ introduced in Section 3 (i.e., applying the evaluation to the derived trees of Figure 11 yields the corresponding derived trees of Figure 8.

7 Applications

In this section, we will discuss a few applications of our main result. Those range from representational issues to algorithmic problems. Finally, we also present a tree transducer model that includes explicit substitution. Such a model might help to address algorithmic problems because derivation and evaluation are intertwined in the model and not separate as in our main result.

7.1 Toolkits

Obviously, our characterization can be applied in a toolkit for extended tree transducers (or STSG) such as TIBURON by May and Knight (2006) to simulate STAG. The existing infrastructure (input-output, derivation mechanism, etc) for extended tree transducers can be re-used to run XTTs encoding STAGs. The only additional overhead is the implementation of the evaluation, which is a straightforward recursive function (as defined in Section 3). After that any STAG can be simulated in the existing framework, which allows experiments with STAG and an evaluation of their expressive power without the need to develop a new toolkit. It should be remarked that some essential algorithms that are very sensitive to the input and output behavior (such as parsing) cannot be simulated by the corresponding algorithms for STSG. It remains an open problem whether the close relationship can also be exploited for such algorithms.

7.2 Algorithms

We already mentioned in the previous section that some algorithms do not easily translate from STAG to STSG (or vice versa) with the help of our characterization. However, many standard algorithms for STAG can easily be derived from the corresponding algorithms for STSG. The simplest example is the union of two STAG. Instead of taking the union of two STAG using the classical construction, we can take the union of the corresponding XTT (or STSG) that simulate the STAGs. Their union will simulate the union of the STAGs. Such properties are especially valuable when we simulate STAG in toolkits for XTT.

A second standard algorithm that easily translates is the algorithm computing the n-best derivations (Huang and Chiang, 2005). Clearly, the n-best derivation algorithm does not consider a particular input or output tree. Since the derivations of the XTT match the derivations of the STAG (in the former the input and output are encoded using explicit substitution), the n-best derivations will coincide. If we are additionally interested in the input and output trees for those n-best derivations, then we can simply evaluate the coded input and output trees returned by n-best derivation algorithm.

Finally, let us consider an algorithm that can be obtained for STAG by developing it for XTT using explicit substitution. We will develop a BAR-HILLEL (Bar-Hillel et al., 1964) construction for STAG. Thus, given a STAG G and a recognizable tree language L, we want to construct a STAG G' such that

$$\tau_{G'} = \{(t, u) \mid (t, u) \in \tau_G, t \in L\} \ .$$

In other words, we take the tree transformation τ_G but additionally require the input tree to be in L. Consequently, this operation is also called *input restriction*. Since STAG are symmetric, the corresponding output restriction can be obtained in the same manner. Note that a classical BAR-HILLEL construction restricting to a regular set of yields can be obtained easily as a particular input restriction. As in Nederhof (2009) a change of model is beneficial for the development of such an algorithm, so we will develop an input restriction for XTT using explicit substitution.

Let $M = (Q, \Sigma, \Delta, I, R)$ be an XTT (using explicit substitution) and $G = (N, \Sigma, I', P)$ be a tree substitution grammar (regular tree grammar) in normal form that recognizes L (i.e., $L(G) = L$). Let $S = \{A \in \Sigma \mid \cdot[\cdot]_A \in \Sigma\}$. A *context* is a mapping $c\colon S \to N$, which remembers a nonterminal of G for each substitution point. Given a rule

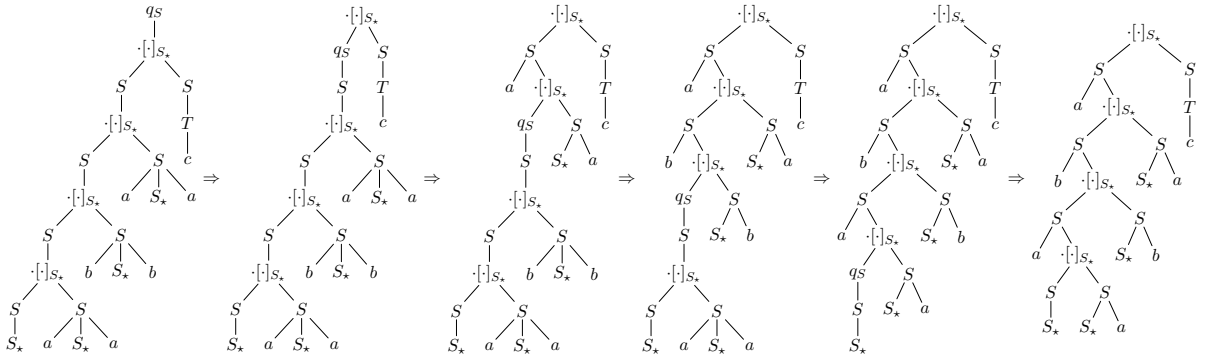


Figure 11: Complete derivation using the constructed XTT rules.

$(q, l) \to (q_1 \cdots q_k, r) \in R$, a nonterminal $p \in N$, and a context $c \in S$, we construct new rules corresponding to successful parses of l subject to the following restrictions:

- If $l = \cdot[\cdot]_A(l_1, l_2)$ for some $A \in \Sigma$, then select $p' \in N$, parse l_1 in p with context c' where $c' = c[A \mapsto p']$[9], and parse l_2 in p' with context c.
- If $l = A_\star$ with $A \in \Sigma$, then $p = c(A)$.
- Finally, if $l = \sigma(l_1, \ldots, l_k)$ for some $\sigma \in \Sigma$, then select $p \to \sigma(p_1, \ldots, p_k) \in P$ is a production of G and we parse l_i with nonterminal p_i and context c for each $1 \leq i \leq k$.

7.3 A complete tree transducer model

So far, we have specified a tree transducer model that requires some additional parsing before it can be applied. This parsing step has to annotate (and correspondingly restructure) the input tree by the adjunction points. This is best illustrated by the left tree in the last pair of trees in Figure 8. To run our constructed XTT on the trivially completed version of this input tree, it has to be transformed into the first tree of Figure 11, where the adjunctions are now visible. In fact, a second un-parsing step is required to evaluate the output.

To avoid the first additional parsing step, we will now modify our tree transducer model such that this parsing step is part of its semantics. This shows that it can also be done locally (instead of globally parsing the whole input tree). In addition, we arrive at a tree transducer model that exactly (up to a relabeling) matches the power of STAG, which can be useful for certain constructions. It is known that an embedded tree transducer (Shieber, 2006) can handle the mentioned un-parsing step.

An *extended embedded tree transducer with substitution* $M = (Q, \Sigma, \Delta, I, R)$ is simply an embedded tree transducer with extended left-hand sides (i.e., any number of input symbols is allowed in the left-hand side) that uses the special symbols $\cdot[\cdot]_A$ in the input. Formally, let

- $Q = Q_0 \cup Q_1$ be finite where Q_0 and Q_1 are the set of states that do not and do have a context parameter, respectively,
- Σ and Δ be ranked alphabets such that if $\cdot[\cdot]_A \in \Sigma$, then $A, A_\star \in \Sigma$,
- $Q\langle U \rangle$ be such that

$$Q\langle U \rangle = \{q\langle u \rangle \mid q \in Q_1, u \in U\} \cup \cup \{q\langle\rangle \mid q \in Q_0\} \ ,$$

- $I \subseteq Q\langle T_\Delta \rangle$, and
- R is a finite set of rules $l \to r$ such that there exists $k \geq 0$ with $l \in Q\langle\{y\}\rangle(C_\Sigma(X_k))$ and $r \in \mathrm{Rhs}_k$ where

$$\mathrm{Rhs}_k := \delta(\mathrm{Rhs}_k, \ldots, \mathrm{Rhs}_k) \mid \mid q_1\langle \mathrm{Rhs}_k \rangle(x) \mid q_0\langle\rangle(x)$$

with $\delta \in \Delta_k$, $q_1 \in Q_1$, $q_0 \in Q_0$, and $x \in X_k$. Moreover, each variable of l (including y) is supposed to occur exactly once in r.

We refer to Shieber (2006) for a full description of embedded tree transducers. As seen from the syntax, we write the context parameter y of a state $q \in Q_1$ as $q\langle y \rangle$. If $q \in Q_0$, then we also write $q\langle\rangle$ or $q\langle\varepsilon\rangle$. In each right-hand side, such a context parameter u can contain output symbols and further calls to input subtrees. The semantics of extended embedded tree transducers with substitution deviates slightly from the embedded tree transducer semantics. Roughly speaking, not its rules as such, but rather their evaluation are now applied in a term-rewrite fashion. Let

$$\mathrm{SF}' := \delta(\mathrm{SF}', \ldots, \mathrm{SF}') \mid \mid q_1\langle \mathrm{SF}' \rangle(t) \mid q_0\langle\rangle(t)$$

[9] c' is the same as c except that it maps A to p'.

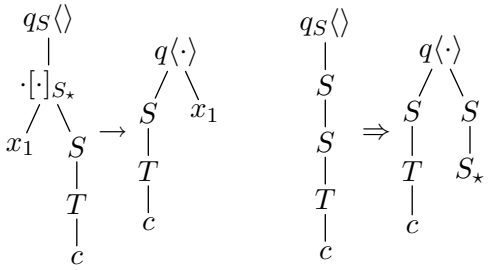


Figure 12: Rule and derivation step using the rule in an extended embedded tree transducer with substitution where the context parameter (if present) is displayed as first child.

where $\delta \in \Delta_k$, $q_1 \in Q_1$, $q_0 \in Q_0$, and $t \in T_\Sigma$.

Given $\xi, \zeta \in \mathrm{SF}'$, we write $\xi \Rightarrow \zeta$ if there exist $C \in C_\Delta(X_1)$, $t_1, \ldots, t_k \in T_\Sigma$, $u \in T_\Delta \cup \{\varepsilon\}$, and a rule $q\langle u\rangle(l) \to r \in R$[10] with $l \in C_\Sigma(X_k)$ such that

- $\xi = C[q\langle u\rangle(l[t_1, \ldots, t_k]^{\mathrm{E}})]$ and
- $\zeta = C[(r[t_1, \ldots, t_k])[u]_y]$.

Note that the essential difference to the "standard" semantics of embedded tree transducers is the evaluation in the first item. The tree transformation computed by M is defined as usual. We illustrate a derivation step in Figure 12, where the match $\cdot[\cdot]_{S_\star}(x_1, S(T(c)))^{\mathrm{E}} = S(S(T(c)))$ is successful for $x_1 = S(S_\star)$.

Theorem 2 *Every STAG can be simulated by an extended embedded tree transducer with substitution. Moreover, every extended embedded tree transducer computes a tree transformation that can be computed by a STAG up to a relabeling.*

8 Conclusions

We presented an alternative view on STAG using tree transducers (or equivalently, STSG). Our main result shows that the syntactic characterization of STAG as STSG plus adjunction rules also carries over to the semantic side. A STAG tree transformation can also be computed by an STSG using explicit substitution. In the light of this result, some standard problems for STAG can be reduced to the corresponding problems for STSG. This allows us to re-use existing algorithms for STSG also for STAG. Moreover, existing STAG algorithms can be related to the corresponding STSG algorithms, which provides further evidence of the close relationship between the two models. We used this relationship to develop a BAR-HILLEL construction for STAG. Finally, we hope that the alternative characterization is easier to handle and might provide further insight into general properties of STAG such as compositions and preservation of regularity.

[10] Note that u is ε if $q \in Q_0$.

Acknowledgements

ANDREAS MALETTI was financially supported by the *Ministerio de Educación y Ciencia* (MEC) grant JDCI-2007-760.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall.

André Arnold and Max Dauchet. 1982. Morphismes et bimorphismes d'arbres. *Theoret. Comput. Sci.*, 20(1):33–93.

Yehoshua Bar-Hillel, Micha Perles, and Eliyahu Shamir. 1964. On formal properties of simple phrase structure grammars. In Yehoshua Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison Wesley.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270. Association for Computational Linguistics.

David Chiang. 2006. An introduction to synchronous grammars. In *Proc. ACL*. Association for Computational Linguistics. Part of a tutorial given with Kevin Knight.

Bruno Courcelle and Paul Franchi-Zannettacci. 1982. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191, 235–257.

Joost Engelfriet and Heiko Vogler. 1985. Macro tree transducers. *J. Comput. System Sci.*, 31(1):71–146.

Ferenc Gécseg and Magnus Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *HLT-NAACL*, pages 105–112. Association for Computational Linguistics. See also (Graehl et al., 2008).

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

Jonathan Graehl, Mark Hopkins, Kevin Knight, and Andreas Maletti. 2009. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proc. IWPT*, pages 53–64. Association for Computational Linguistics.

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proc. CICLing*, volume 3406 of LNCS, pages 1–24. Springer.

Kevin Knight. 2007. Capturing practical natural language transformations. *Machine Translation*, 21(2):121–133.

Andreas Maletti. 2008. Compositions of extended top-down tree transducers. *Inform. and Comput.*, 206(9–10):1187–1196.

Jonathan May and Kevin Knight. 2006. TIBURON: A weighted tree automata toolkit. In *Proc. CIAA*, volume 4094 of LNCS, pages 102–113. Springer.

Mark-Jan Nederhof. 2009. Weighted parsing of trees. In *Proc. IWPT*, pages 13–24. Association for Computational Linguistics.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Optimal k-arization of synchronous tree-adjoining grammar. In *Proc. ACL*, pages 604–612. Association for Computational Linguistics.

William C. Rounds. 1970. Mappings and grammars on trees. *Math. Systems Theory*, 4(3):257–287.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. Computational Linguistics*, volume 3, pages 253–258.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proc. TAG+7*, pages 88–95.

Stuart M. Shieber. 2006. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proc. EACL*, pages 377–384. Association for Computational Linguistics.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. Workshop on Syntax and Structure in Statistical Translation*, pages 88–95. Association for Computational Linguistics.

James W. Thatcher. 1970. Generalized^2 sequential machine maps. *J. Comput. System Sci.*, 4(4):339–367.

Dynamic Programming for Linear-Time Incremental Parsing

Liang Huang
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
lhuang@isi.edu

Kenji Sagae
USC Institute for Creative Technologies
13274 Fiji Way
Marina del Rey, CA 90292
sagae@ict.usc.edu

Abstract

Incremental parsing techniques such as shift-reduce have gained popularity thanks to their efficiency, but there remains a major problem: the search is *greedy* and only explores a tiny fraction of the whole space (even with beam search) as opposed to dynamic programming. We show that, surprisingly, dynamic programming is in fact possible for many shift-reduce parsers, by merging "equivalent" stacks based on feature values. Empirically, our algorithm yields up to a five-fold speedup over a state-of-the-art shift-reduce dependency parser with no loss in accuracy. Better search also leads to better learning, and our final parser outperforms all previously reported dependency parsers for English and Chinese, yet is much faster.

1 Introduction

In terms of search strategy, most parsing algorithms in current use for data-driven parsing can be divided into two broad categories: *dynamic programming* which includes the dominant CKY algorithm, and *greedy search* which includes most incremental parsing methods such as shift-reduce.[1] Both have pros and cons: the former performs an *exact* search (in cubic time) over an exponentially large space, while the latter is much faster (in linear-time) and is psycholinguistically motivated (Frazier and Rayner, 1982), but its greedy nature may suffer from severe search errors, as it only explores a tiny fraction of the whole space even with a beam.

Can we combine the advantages of both approaches, that is, construct an incremental parser that runs in (almost) linear-time, yet searches over a huge space with dynamic programming?

Theoretically, the answer is negative, as Lee (2002) shows that context-free parsing can be used to compute matrix multiplication, where sub-cubic algorithms are largely impractical.

We instead propose a dynamic programming alogorithm for shift-reduce parsing which runs in polynomial time in theory, but linear-time (with beam search) in practice. The key idea is to merge equivalent stacks according to feature functions, inspired by Earley parsing (Earley, 1970; Stolcke, 1995) and generalized LR parsing (Tomita, 1991). However, our formalism is more flexible and our algorithm more practical. Specifically, we make the following contributions:

- *theoretically*, we show that for a large class of modern shift-reduce parsers, dynamic programming is in fact possible and runs in polynomial time as long as the feature functions are *bounded* and *monotonic* (which almost always holds in practice);
- *practically*, dynamic programming is up to five times faster (with the same accuracy) as conventional beam-search on top of a state-of-the-art shift-reduce dependency parser;
- as a by-product, dynamic programming can output a *forest* encoding exponentially many trees, out of which we can draw better and longer k-best lists than beam search can;
- finally, better and faster search also leads to better and faster learning. Our final parser achieves the best (unlabeled) accuracies that we are aware of in both English and Chinese among dependency parsers trained on the Penn Treebanks. Being linear-time, it is also much faster than most other parsers, even with a pure Python implementation.

[1] McDonald et al. (2005b) is a notable exception: the MST algorithm is exact search but not dynamic programming.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1077–1086,
Uppsala, Sweden, 11-16 July 2010.

input: $w_0 \ldots w_{n-1}$

axiom $0 : \langle 0,\ \epsilon \rangle : 0$

$$\textsf{sh} \quad \frac{\ell : \langle j,\ S \rangle : c}{\ell + 1 : \langle j+1,\ S|w_j \rangle : c + \xi} \ j < n$$

$$\textsf{re}_{\curvearrowleft} \quad \frac{\ell : \langle j,\ S|s_1|s_0 \rangle : c}{\ell + 1 : \langle j,\ S|s_1{}^{\curvearrowleft}s_0 \rangle : c + \lambda}$$

$$\textsf{re}_{\curvearrowright} \quad \frac{\ell : \langle j,\ S|s_1|s_0 \rangle : c}{\ell + 1 : \langle j,\ S|s_1{}^{\curvearrowright}s_0 \rangle : c + \rho}$$

goal $2n - 1 : \langle n,\ s_0 \rangle : c$

where ℓ is the step, c is the cost, and the shift cost ξ and reduce costs λ and ρ are:

$$\xi = \mathbf{w} \cdot \mathbf{f}_{\textsf{sh}}(j, S) \quad (1)$$
$$\lambda = \mathbf{w} \cdot \mathbf{f}_{\textsf{re}_{\curvearrowleft}}(j, S|s_1|s_0) \quad (2)$$
$$\rho = \mathbf{w} \cdot \mathbf{f}_{\textsf{re}_{\curvearrowright}}(j, S|s_1|s_0) \quad (3)$$

Figure 1: Deductive system of vanilla shift-reduce.

For convenience of presentation and experimentation, we will focus on shift-reduce parsing for dependency structures in the remainder of this paper, though our formalism and algorithm can also be applied to phrase-structure parsing.

2 Shift-Reduce Parsing

2.1 Vanilla Shift-Reduce

Shift-reduce parsing performs a left-to-right scan of the input sentence, and at each step, choose one of the two actions: either *shift* the current word onto the stack, or *reduce* the top two (or more) items at the end of the stack (Aho and Ullman, 1972). To adapt it to dependency parsing, we split the reduce action into two cases, $\textsf{re}_{\curvearrowleft}$ and $\textsf{re}_{\curvearrowright}$, depending on which one of the two items becomes the head after reduction. This procedure is known as "arc-standard" (Nivre, 2004), and has been engineered to achieve state-of-the-art parsing accuracy in Huang et al. (2009), which is also the reference parser in our experiments.[2]

More formally, we describe a parser configuration by a *state* $\langle j, S \rangle$ where S is a stack of trees $s_0, s_1, \ldots$ where s_0 is the top tree, and j is the queue head position (current word q_0 is w_j). At each step, we choose one of the three actions:

1. sh: move the head of queue, w_j, onto stack S as a singleton tree;
2. $\textsf{re}_{\curvearrowleft}$: combine the top two trees on the stack, s_0 and s_1, and replace them with tree $s_1{}^{\curvearrowleft}s_0$.
3. $\textsf{re}_{\curvearrowright}$: combine the top two trees on the stack, s_0 and s_1, and replace them with tree $s_1{}^{\curvearrowright}s_0$.

Note that the shorthand notation $t^{\curvearrowleft}t'$ denotes a new tree by "attaching tree t' as the leftmost child of the root of tree t". This procedure can be summarized as a deductive system in Figure 1. States are organized according to step ℓ, which denotes the number of actions accumulated. The parser runs in linear-time as there are exactly $2n-1$ steps for a sentence of n words.

As an example, consider the sentence *"I saw Al with Joe"* in Figure 2. At step (4), we face a shift-reduce conflict: either combine "saw" and "Al" in a $\textsf{re}_{\curvearrowright}$ action (5a), or shift "with" (5b). To resolve this conflict, there is a **cost** c associated with each state so that we can pick the best one (or few, with a beam) at each step. Costs are accumulated in each step: as shown in Figure 1, actions sh, $\textsf{re}_{\curvearrowleft}$, and $\textsf{re}_{\curvearrowright}$ have their respective costs ξ, λ, and ρ, which are dot-products of the weights $\mathbf{w}$ and *features* extracted from the state and the action.

input: "I saw Al with Joe"

step	*action*	*stack*	*queue*
0	-		I ...
1	sh	I	saw ...
2	sh	I saw	Al ...
3	$\textsf{re}_{\curvearrowleft}$	I$^{\curvearrowleft}$saw	Al ...
4	sh	I$^{\curvearrowleft}$saw Al	with ...
5a	$\textsf{re}_{\curvearrowright}$	I$^{\curvearrowleft}$saw$^{\curvearrowright}$Al	with ...
5b	sh	I$^{\curvearrowleft}$saw Al with	Joe

Figure 2: A trace of vanilla shift-reduce. After step (4), the parser branches off into (5a) or (5b).

2.2 Features

We view features as "abstractions" or (partial) observations of the current state, which is an important intuition for the development of dynamic programming in Section 3. **Feature templates** are functions that draw information from the *feature window* (see Tab. 1(b)), consisting of the top few trees on the stack and the first few words on the queue. For example, one such feature template $f_{100} = s_0.\mathsf{w} \circ q_0.\mathsf{t}$ is a conjunction

[2] There is another popular variant, "arc-eager" (Nivre, 2004; Zhang and Clark, 2008), which is more complicated and less similar to the classical shift-reduce algorithm.

of two **atomic features** s_0.w and q_0.t, capturing the root word of the top tree s_0 on the stack, and the part-of-speech tag of the current head word q_0 on the queue. See Tab. 1(a) for the list of feature templates used in the full model. Feature templates are instantiated for a specific state. For example, at step (4) in Fig. 2, the above template f_{100} will generate a feature instance

$$(s_0.\mathsf{w} = \text{Al}) \circ (q_0.\mathsf{t} = \text{IN}).$$

More formally, we denote $\mathbf{f}$ to be the **feature function**, such that $\mathbf{f}(j, S)$ returns a vector of feature instances for state $\langle j, S\rangle$. To decide which action is the best for the current state, we perform a three-way classification based on $\mathbf{f}(j, S)$, and to do so, we further conjoin these feature instances with the action, producing action-conjoined instances like

$$(s_0.\mathsf{w} = \text{Al}) \circ (q_0.\mathsf{t} = \text{IN}) \circ (action = \mathsf{sh}).$$

We denote $\mathbf{f}_{\mathsf{sh}}(j, S)$, $\mathbf{f}_{\mathsf{re}_\frown}(j, S)$, and $\mathbf{f}_{\mathsf{re}_\smile}(j, S)$ to be the conjoined feature instances, whose dot-products with the weight vector decide the best action (see Eqs. (1-3) in Fig. 1).

2.3 Beam Search and Early Update

To improve on strictly greedy search, shift-reduce parsing is often enhanced with beam search (Zhang and Clark, 2008), where b states develop in parallel. At each step we extend the states in the current beam by applying one of the three actions, and then choose the best b resulting states for the next step. Our dynamic programming algorithm also runs on top of beam search in practice.

To train the model, we use the averaged perceptron algorithm (Collins, 2002). Following Collins and Roark (2004) we also use the "early-update" strategy, where an update happens whenever the gold-standard action-sequence falls off the beam, with the rest of the sequence neglected.[3] The intuition behind this strategy is that later mistakes are often caused by previous ones, and are irrelevant when the parser is on the wrong track. Dynamic programming turns out to be a great fit for early updating (see Section 4.3 for details).

3 Dynamic Programming (DP)

3.1 Merging Equivalent States

The key observation for dynamic programming is to merge "equivalent states" in the same beam (i.e., same step) if they have the same feature values, because they will have the same costs as shown in the deductive system in Figure 1. Thus we can define two states $\langle j, S\rangle$ and $\langle j', S'\rangle$ to be equivalent, notated $\langle j, S\rangle \sim \langle j', S'\rangle$, iff.

$$j = j' \quad \text{and} \quad \mathbf{f}(j, S) = \mathbf{f}(j', S'). \qquad (4)$$

Note that $j = j'$ is also needed because the queue head position j determines which word to shift next. In practice, however, a small subset of atomic features will be enough to determine the whole feature vector, which we call **kernel features** $\widetilde{\mathbf{f}}(j, S)$, defined as the *smallest set* of atomic templates such that

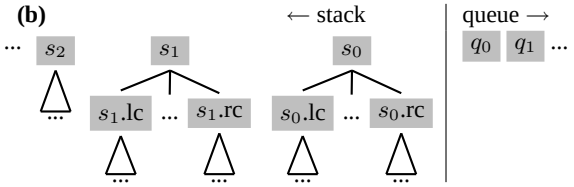
$$\widetilde{\mathbf{f}}(j, S) = \widetilde{\mathbf{f}}(j', S') \Rightarrow \langle j, S\rangle \sim \langle j', S'\rangle.$$

For example, the full list of 28 feature templates in Table 1(a) can be determined by just 12 atomic features in Table 1(c), which just look at the root words and tags of the top two trees on stack, as well as the tags of their left- and rightmost children, plus the root tag of the third tree s_2, and finally the word and tag of the queue head q_0 and the

(a) **Feature Templates $\mathbf{f}(j, S)$** $q_i = w_{j+i}$

(1)	s_0.w	s_0.t	s_0.w ∘ s_0.t
	s_1.w	s_1.t	s_1.w ∘ s_1.t
	q_0.w	q_0.t	q_0.w ∘ q_0.t
(2)	s_0.w ∘ s_1.w		s_0.t ∘ s_1.t
	s_0.t ∘ q_0.t		s_0.w ∘ s_0.t ∘ s_1.t
	s_0.t ∘ s_1.w ∘ s_1.t		s_0.w ∘ s_1.w ∘ s_1.t
	s_0.w ∘ s_0.t ∘ s_1.w		s_0.w ∘ s_0.t ∘ s_1 ∘ s_1.t
(3)	s_0.t ∘ q_0.t ∘ q_1.t		s_1.t ∘ s_0.t ∘ q_0.t
	s_0.w ∘ q_0.t ∘ q_1.t		s_1.t ∘ s_0.w ∘ q_0.t
(4)	s_1.t ∘ s_1.lc.t ∘ s_0.t		s_1.t ∘ s_1.rc.t ∘ s_0.t
	s_1.t ∘ s_0.t ∘ s_0.rc.t		s_1.t ∘ s_1.lc.t ∘ s_0
	s_1.t ∘ s_1.rc.t ∘ s_0.w		s_1.t ∘ s_0.w ∘ s_0.lc.t
(5)	s_2.t ∘ s_1.t ∘ s_0.t		

(b)



(c) **Kernel features for DP** $\widetilde{\mathbf{f}}(j, S) = (j, \mathbf{f}_2(s_2), \mathbf{f}_1(s_1), \mathbf{f}_0(s_0))$

$\mathbf{f}_2(s_2)$	s_2.t			
$\mathbf{f}_1(s_1)$	s_1.w	s_1.t	s_1.lc.t	s_1.rc.t
$\mathbf{f}_0(s_0)$	s_0.w	s_0.t	s_0.lc.t	s_0.rc.t
j	q_0.w	q_0.t	q_1.t	

Table 1: **(a)** feature templates used in this work, adapted from Huang et al. (2009). x.w and x.t denotes the root word and POS tag of tree (or word) x. and x.lc and x.rc denote x's left- and rightmost child. **(b)** feature window. **(c)** kernel features.

[3] As a special case, for the deterministic mode (b=1), updates always co-occur with the first mistake made.

state form $\ell : \langle i,\ j,\ s_d...s_0\rangle$: (c, v, π) — ℓ: step; c, v: prefix and inside costs; π: predictor states

equivalence $\ell : \langle i,\ j,\ s_d...s_0\rangle \sim \ell : \langle i,\ j,\ s'_d...s'_0\rangle$ iff. $\widetilde{\mathbf{f}}(j, s_d...s_0) = \widetilde{\mathbf{f}}(j, s'_d...s'_0)$

ordering $\ell : \_ :\ (c, v, \_) \prec \ell : \_ :\ (c', v', \_)$ iff. $c < c'$ or ($c = c'$ and $v < v'$).

---

axiom (p_0)
$$0 : \langle 0,\ 0,\ \epsilon\rangle :\ (0, 0, \emptyset)$$

sh
$$\frac{\overset{\text{state } p\text{:}}{\ell : \langle \_,\ j,\ s_d...s_0\rangle :\ (c, \_, \_)}}{\ell + 1 : \langle j,\ j+1,\ s_{d-1}...s_0, w_j\rangle :\ (c + \xi,\ 0,\ \{p\})}\ j < n$$

re$_\curvearrowright$
$$\frac{\overset{\text{state } p\text{:}}{\_ : \langle k,\ i,\ s'_d...s'_0\rangle :\ (c', v', \pi')} \qquad \overset{\text{state } q\text{:}}{\ell : \langle i,\ j,\ s_d...s_0\rangle :\ (\_, v, \pi)}}{\ell + 1 : \langle k,\ j,\ s'_d...s'_1, s'_0 \curvearrowright s_0\rangle :\ (c' + v + \delta,\ v' + v + \delta,\ \pi')}\ p \in \pi$$

goal
$$2n - 1 : \langle 0,\ n,\ s_d...s_0\rangle :\ (c,\ c, \{p_0\})$$

where $\xi = \mathbf{w} \cdot \mathbf{f}_{\mathsf{sh}}(j, s_d...s_0)$, and $\delta = \xi' + \lambda$, with $\xi' = \mathbf{w} \cdot \mathbf{f}_{\mathsf{sh}}(i, s'_d...s'_0)$ and $\lambda = \mathbf{w} \cdot \mathbf{f}_{\mathsf{re}_\curvearrowright}(j, s_d...s_0)$.

Figure 3: Deductive system for shift-reduce parsing with dynamic programming. The predictor state set π is an implicit graph-structured stack (Tomita, 1988) while the prefix cost c is inspired by Stolcke (1995). The re$_\curvearrowright$ case is similar, replacing $s'_0 \curvearrowleft s_0$ with $s'_0 \curvearrowright s_0$, and λ with $\rho = \mathbf{w} \cdot \mathbf{f}_{\mathsf{re}_\curvearrowright}(j, s_d...s_0)$. Irrelevant information in a deduction step is marked as an underscore (_) which means "can match anything".

tag of the next word q_1. Since the queue is *static information* to the parser (unlike the stack, which changes dynamically), we can use j to replace features from the queue. So in general we write

$$\widetilde{\mathbf{f}}(j, S) = (j, \mathbf{f}_d(s_d), \ldots, \mathbf{f}_0(s_0))$$

if the feature window looks at top $d + 1$ trees on stack, and where $\mathbf{f}_i(s_i)$ extracts kernel features from tree s_i ($0 \le i \le d$). For example, for the full model in Table 1(a) we have

$$\widetilde{\mathbf{f}}(j, S) = (j, \mathbf{f}_2(s_2), \mathbf{f}_1(s_1), \mathbf{f}_0(s_0)), \qquad (5)$$

where $d = 2$, $\mathbf{f}_2(x) = x.\mathsf{t}$, and $\mathbf{f}_1(x) = \mathbf{f}_0(x) = (x.\mathsf{w}, x.\mathsf{t}, x.\mathsf{lc.t}, x.\mathsf{rc.t})$ (see Table 1(c)).

3.2 Graph-Structured Stack and Deduction

Now that we have the kernel feature functions, it is intuitive that we might only need to remember the relevant bits of information from only the *last* $(d + 1)$ *trees* on stack instead of the whole stack, because they provide all the relevant information for the features, and thus determine the costs. For shift, this suffices as the stack grows on the right; but for reduce actions the stack shrinks, and in order still to maintain $d + 1$ trees, we have to know something about the history. This is exactly why we needed the full stack for vanilla shift-reduce parsing in the first place, and why dynamic programming seems hard here.

To solve this problem we borrow the idea of "graph-structured stack" (GSS) from Tomita (1991). Basically, each state p carries with it a set $\pi(p)$ of **predictor states**, each of which can be combined with p in a reduction step. In a shift step, if state p generates state q (we say "p predicts q" in Earley (1970) terms), then p is added onto $\pi(q)$. When two equivalent shifted states get merged, their predictor states get combined. In a reduction step, state q tries to combine with every predictor state $p \in \pi(q)$, and the resulting state r inherits the predictor states set from p, i.e., $\pi(r) = \pi(p)$. Interestingly, when two equivalent reduced states get merged, we can prove (by induction) that their predictor states are identical (proof omitted).

Figure 3 shows the new deductive system with dynamic programming and GSS. A new state has the form

$$\ell : \langle i,\ j,\ s_d...s_0\rangle$$

where $[i..j]$ is the span of the top tree s_0, and $s_d..s_1$ are merely "left-contexts". It can be combined with some predictor state p spanning $[k..i]$

$$\ell' : \langle k,\ i,\ s'_d...s'_0\rangle$$

to form a larger state spanning $[k..j]$, with the resulting top tree being either $s_1 \curvearrowleft s_0$ or $s_1 \curvearrowright s_0$.

This style resembles CKY and Earley parsers. In fact, the *chart* in Earley and other agenda-based parsers is indeed a GSS when viewed left-to-right. In these parsers, when a state is popped up from the agenda, it looks for possible sibling states that can combine with it; GSS, however, *explicitly* maintains these predictor states so that the newly-popped state does not need to look them up.[4]

3.3 Correctness and Polynomial Complexity

We state the main theoretical result with the proof omitted due to space constraints:

Theorem 1. *The deductive system is optimal and runs in worst-case polynomial time as long as the kernel feature function satisfies two properties:*

- **bounded***:* $\widetilde{\mathbf{f}}(j, S) = (j, \mathbf{f}_d(s_d), \ldots, \mathbf{f}_0(s_0))$ *for some constant* d*, and each* $|\mathbf{f}_t(x)|$ *also bounded by a constant for all possible tree* x*.*
- **monotonic***:* $\mathbf{f}_t(x) = \mathbf{f}_t(y) \Rightarrow \mathbf{f}_{t+1}(x) = \mathbf{f}_{t+1}(y)$*, for all* t *and all possible trees* x*,* y*.*

Intuitively, boundedness means features can only look at a local window and can only extract bounded information on each tree, which is always the case in practice since we can not have infinite models. Monotonicity, on the other hand, says that features drawn from trees farther away from the top should *not* be more refined than from those closer to the top. This is also natural, since the information most relevant to the current decision is always around the stack top. For example, the kernel feature function in Eq. 5 is bounded and monotonic, since $\mathbf{f}_2$ is less refined than $\mathbf{f}_1$ and $\mathbf{f}_0$.

These two requirements are related to grammar refinement by annotation (Johnson, 1998), where annotations must be bounded and monotonic: for example, one cannot refine a grammar by only remembering the grandparent but not the parent symbol. The difference here is that the annotations are not vertical ((grand-)parent), but rather *horizontal* (left context). For instance, a context-free rule $A \rightarrow B\ C$ would become ${}^{D}A \rightarrow {}^{D}B\ {}^{B}C$ for some D if there exists a rule $E \rightarrow \alpha D A \beta$. This resembles the reduce step in Fig. 3.

The very high-level idea of the proof is that boundedness is crucial for polynomial-time, while monotonicity is used for the optimal substructure property required by the correctness of DP.

[4] In this sense, GSS (Tomita, 1988) is really not a new invention: an efficient implementation of Earley (1970) should already have it implicitly, similar to what we have in Fig. 3.

3.4 Beam Search based on Prefix Cost

Though the DP algorithm runs in polynomial-time, in practice the complexity is still too high, esp. with a rich feature set like the one in Table 1. So we apply the same beam search idea from Sec. 2.3, where each step can accommodate only the best b states. To decide the ordering of states in each beam we borrow the concept of **prefix cost** from Stolcke (1995), originally developed for weighted Earley parsing. As shown in Fig. 3, the prefix cost c is the total cost of the best action sequence from the initial state to the end of state p, i.e., it includes both the **inside cost** v (for Viterbi inside derivation), and the cost of the (best) path leading towards the beginning of state p. We say that a state p with prefix cost c is better than a state p' with prefix cost c', notated $p \prec p'$ in Fig. 3, if $c < c'$. We can also prove (by contradiction) that optimizing for prefix cost implies optimal inside cost (Nederhof, 2003, Sec. 4). [5]

As shown in Fig. 3, when a state q with costs (c, v) is combined with a predictor state p with costs (c', v'), the resulting state r will have costs

$$(c' + v + \delta,\ v' + v + \delta),$$

where the inside cost is intuitively the combined inside costs plus an additional combo cost δ from the combination, while the resulting prefix cost $c' + v + \delta$ is the sum of the prefix cost of the predictor state q, the inside cost of the current state p, and the combo cost. Note the prefix cost of q is irrelevant. The combo cost $\delta = \xi' + \lambda$ consists of shift cost ξ' of p and reduction cost λ of q.

The cost in the non-DP shift-reduce algorithm (Fig. 1) is indeed a prefix cost, and the DP algorithm subsumes the non-DP one as a special case where no two states are equivalent.

3.5 Example: Edge-Factored Model

As a concrete example, Figure 4 simulates an edge-factored model (Eisner, 1996; McDonald et al., 2005a) using shift-reduce with dynamic programming, which is similar to bilexical PCFG parsing using CKY (Eisner and Satta, 1999). Here the kernel feature function is

$$\widetilde{\mathbf{f}}(j, S) = (j, h(s_1), h(s_0))$$

[5] Note that using inside cost v for ordering would be a bad idea, as it will always prefer shorter derivations like in best-first parsing. As in A* search, we need some estimate of "outside cost" to predict which states are more promising, and the prefix cost includes an exact cost for the left outside context, but no right outside context.

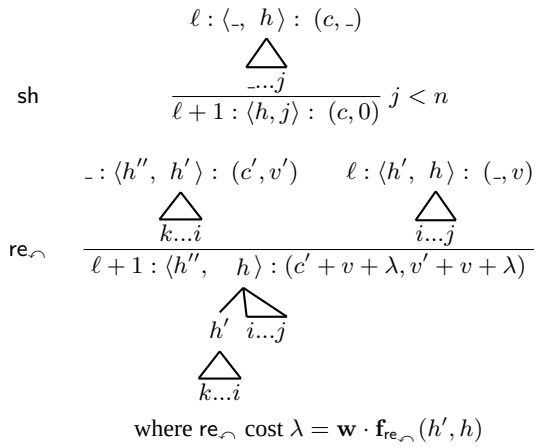


Figure 4: Example of shift-reduce with dynamic programming: simulating an edge-factored model. GSS is implicit here, and re$_\frown$ case omitted.

where $h(x)$ returns the head word index of tree x, because all features in this model are based on the head and modifier indices in a dependency link. This function is obviously bounded and monotonic in our definitions. The theoretical complexity of this algorithm is $O(n^7)$ because in a reduction step we have three span indices and three head indices, plus a step index ℓ. By contrast, the naïve CKY algorithm for this model is $O(n^5)$ which can be improved to $O(n^3)$ (Eisner, 1996).[6] The higher complexity of our algorithm is due to two factors: first, we have to maintain both h and h' in one state, because the current shift-reduce model can not draw features *across* different states (unlike CKY); and more importantly, we group states by step ℓ in order to achieve incrementality and linear runtime with beam search that is not (easily) possible with CKY or MST.

4 Experiments

We first reimplemented the reference shift-reduce parser of Huang et al. (2009) in Python (henceforth "non-DP"), and then extended it to do dynamic programing (henceforth "DP"). We evaluate their performances on the standard Penn Treebank (PTB) English dependency parsing task[7] using the standard split: secs 02-21 for training, 22 for development, and 23 for testing. Both DP and non-DP parsers use the same feature templates in Table 1. For Secs. 4.1-4.2, we use a baseline model trained with non-DP for both DP and non-DP, so that we can do a side-by-side comparison of search quality; in Sec. 4.3 we will retrain the model with DP and compare it against training with non-DP.

4.1 Speed Comparisons

To compare parsing speed between DP and non-DP, we run each parser on the development set, varying the beam width b from 2 to 16 (DP) or 64 (non-DP). Fig. 5a shows the relationship between search quality (as measured by the average model score per sentence, higher the better) and speed (average parsing time per sentence), where DP with a beam width of b=16 achieves the same search quality with non-DP at b=64, while being 5 times faster. Fig. 5b shows a similar comparison for dependency accuracy. We also test with an edge-factored model (Sec. 3.5) using feature templates (1)-(3) in Tab. 1, which is a subset of those in McDonald et al. (2005b). As expected, this difference becomes more pronounced (8 times faster in Fig. 5c), since the less expressive feature set makes more states "equivalent" and mergeable in DP. Fig. 5d shows the (almost linear) correlation between dependency accuracy and search quality, confirming that better search yields better parsing.

4.2 Search Space, Forest, and Oracles

DP achieves better search quality because it expores an exponentially large search space rather than only b trees allowed by the beam (see Fig. 6a). As a by-product, DP can output a *forest* encoding these exponentially many trees, out of which we can draw longer and better (in terms of oracle) k-best lists than those in the beam (see Fig. 6b). The forest itself has an oracle of 98.15 (as if $k \rightarrow \infty$), computed à la Huang (2008, Sec. 4.1). These candidate sets may be used for reranking (Charniak and Johnson, 2005; Huang, 2008).[8]

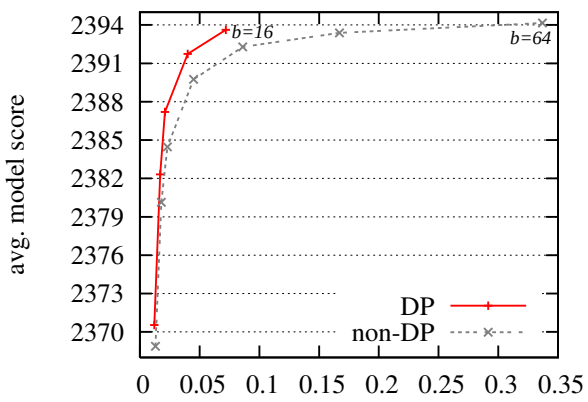
4.3 Perceptron Training and Early Updates

Another interesting advantage of DP over non-DP is the faster training with perceptron, even when both parsers use the same beam width. This is due to the use of early updates (see Sec. 2.3), which happen much more often with DP, because a gold-standard state p is often merged with an equivalent (but incorrect) state that has a higher model score, which triggers update immediately. By contrast, in non-DP beam search, states such as p might still

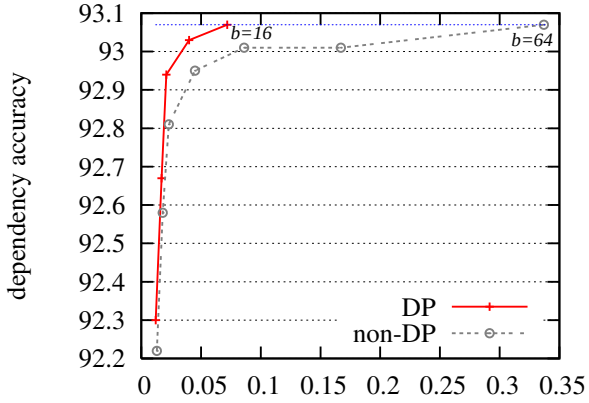
[6] Or $O(n^2)$ with MST, but including non-projective trees.

[7] Using the head rules of Yamada and Matsumoto (2003).

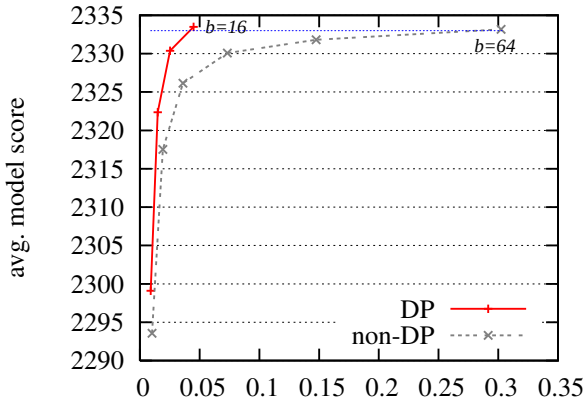
[8] DP's k-best lists are extracted from the forest using the algorithm of Huang and Chiang (2005), rather than those in the final beam as in the non-DP case, because many derivations have been merged during dynamic programming.



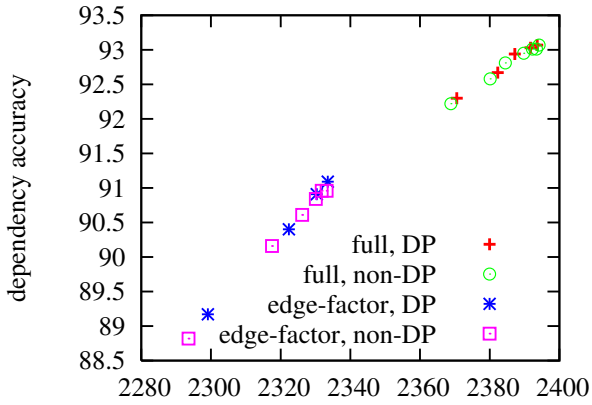
(a) search quality vs. time (full model)



(b) parsing accuracy vs. time (full model)

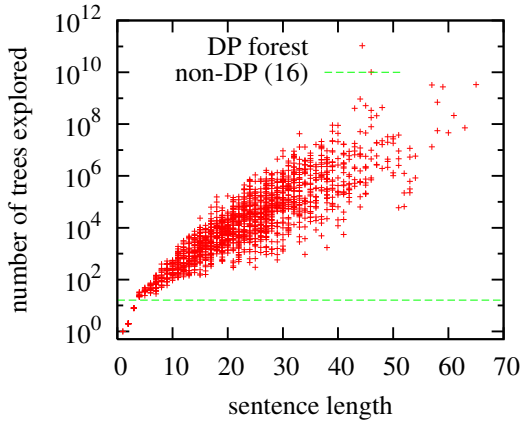


(c) search quality vs. time (edge-factored model)

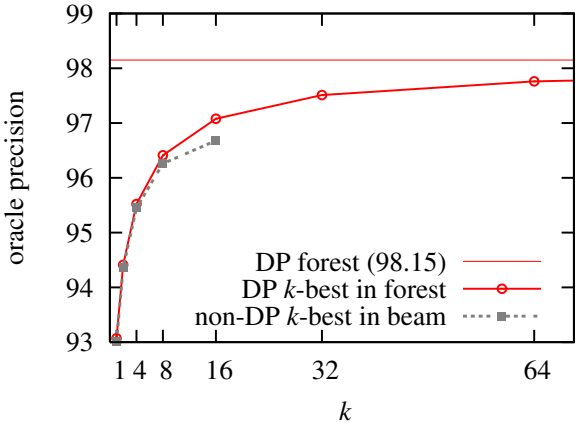


(d) correlation b/w parsing (y) and search (x)

Figure 5: Speed comparisons between DP and non-DP, with beam size b ranging 2∼16 for DP and 2∼64 for non-DP. Speed is measured by avg. parsing time (secs) per sentence on x axis. With the same level of search quality or parsing accuracy, DP (at b=16) is ∼4.8 times faster than non-DP (at b=64) with the full model in plots (a)-(b), or ∼8 times faster with the simplified edge-factored model in plot (c). Plot (d) shows the (roughly linear) correlation between parsing accuracy and search quality (avg. model score).



(a) sizes of search spaces



(b) oracle precision on dev

Figure 6: DP searches over a forest of *exponentially many* trees, which also produces better and longer k-best lists with higher oracles, while non-DP only explores b trees allowed in the beam ($b = 16$ here).

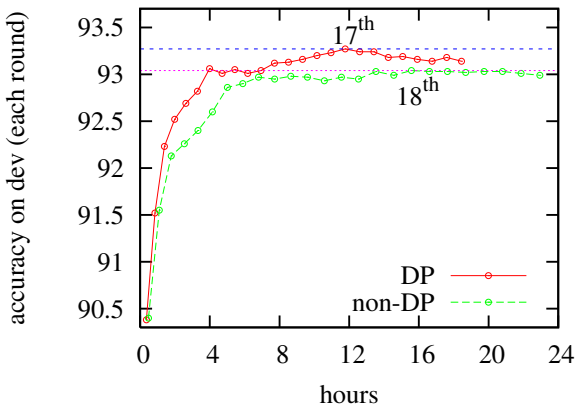


Figure 7: Learning curves (showing precision on dev) of perceptron training for 25 iterations (b=8). DP takes 18 hours, peaking at the 17th iteration (93.27%) with 12 hours, while non-DP takes 23 hours, peaking at the 18th (93.04%) with 16 hours.

survive in the beam throughout, even though it is no longer possible to rank the best in the beam.

The higher frequency of early updates results in faster iterations of perceptron training. Table 2 shows the percentage of early updates and the time per iteration during training. While the number of updates is roughly comparable between DP and non-DP, the rate of early updates is much higher with DP, and the time per iteration is consequently shorter. Figure 7 shows that training with DP is about 1.2 times faster than non-DP, and achieves +0.2% higher accuracy on the dev set (93.27%).

Besides training with gold POS tags, we also trained on noisy tags, since they are closer to the test setting (automatic tags on sec 23). In that case, we tag the dev and test sets using an automatic POS tagger (at 97.2% accuracy), and tag the training set using four-way jackknifing similar to Collins (2000), which contributes another +0.1% improvement in accuracy on the test set. Faster training also enables us to incorporate more features, where we found more lookahead features (q_2) results in another +0.3% improvement.

4.4 Final Results on English and Chinese

Table 3 presents the final test results of our DP parser on the Penn English Treebank, compared with other state-of-the-art parsers. Our parser achieves the highest (unlabeled) dependency accuracy among dependency parsers trained on the Treebank, and is also much faster than most other parsers even with a pure Python implementation

it	update	early%	time	update	early%	time
1	31943	98.9	22	31189	87.7	29
5	20236	98.3	38	19027	70.3	47
17	8683	97.1	48	7434	49.5	60
25	5715	97.2	51	4676	41.2	65

Table 2: Perceptron iterations with DP (left) and non-DP (right). Early updates happen much more often with DP due to equivalent state merging, which leads to faster training (time in minutes).

	word	L	time	comp.
McDonald 05b	90.2	Ja	0.12	$O(n^2)$
McDonald 05a	90.9	Ja	0.15	$O(n^3)$
Koo 08 base	92.0	–	–	$O(n^4)$
Zhang 08 single	91.4	C	0.11	$O(n)^\ddagger$
this work	**92.1**	**Py**	**0.04**	$O(n)$
†Charniak 00	92.5	C	0.49	$O(n^5)$
†Petrov 07	92.4	Ja	0.21	$O(n^3)$
Zhang 08 combo	92.1	C	–	$O(n^2)^\ddagger$
Koo 08 semisup	93.2	–	–	$O(n^4)$

Table 3: Final test results on English (PTB). Our parser (in pure Python) has the highest accuracy among dependency parsers trained on the Treebank, and is also much faster than major parsers. †converted from constituency trees. C=C/C++, Py=Python, Ja=Java. Time is in seconds per sentence. Search spaces: ‡linear; others exponential.

(on a 3.2GHz Xeon CPU). Best-performing constituency parsers like Charniak (2000) and Berkeley (Petrov and Klein, 2007) do outperform our parser, since they consider more information during parsing, but they are at least 5 times slower. Figure 8 shows the parse time in seconds for each test sentence. The observed time complexity of our DP parser is in fact linear compared to the superlinear complexity of Charniak, MST (McDonald et al., 2005b), and Berkeley parsers. Additional techniques such as semi-supervised learning (Koo et al., 2008) and parser combination (Zhang and Clark, 2008) do achieve accuracies equal to or higher than ours, but their results are not directly comparable to ours since they have access to extra information like unlabeled data. Our technique is orthogonal to theirs, and combining these techniques could potentially lead to even better results.

We also test our final parser on the Penn Chinese Treebank (CTB5). Following the set-up of Duan et al. (2007) and Zhang and Clark (2008), we split CTB5 into training (secs 001-815 and 1001-

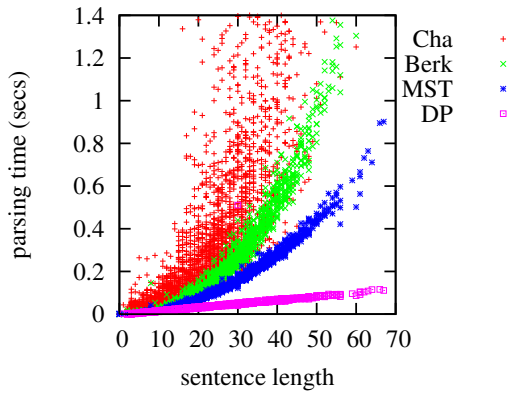


Figure 8: Scatter plot of parsing time against sentence length, comparing with Charniak, Berkeley, and the $O(n^2)$ MST parsers.

	word	non-root	root	compl.
Duan 07	83.88	84.36	73.70	32.70
Zhang 08†	84.33	84.69	76.73	32.79
this work	85.20	85.52	78.32	33.72

Table 4: Final test results on Chinese (CTB5). †The transition parser in Zhang and Clark (2008).

1136), development (secs 886-931 and 1148-1151), and test (secs 816-885 and 1137-1147) sets, assume gold-standard POS-tags for the input, and use the head rules of Zhang and Clark (2008). Table 4 summarizes the final test results, where our work performs the best in all four types of (unlabeled) accuracies: word, non-root, root, and complete match (all excluding punctuations). [9,10]

5 Related Work

This work was inspired in part by Generalized LR parsing (Tomita, 1991) and the graph-structured stack (GSS). Tomita uses GSS for exhaustive LR parsing, where the GSS is equivalent to a dynamic programming chart in chart parsing (see Footnote 4). In fact, Tomita's GLR is an instance of techniques for tabular simulation of nondeterministic pushdown automata based on deductive systems (Lang, 1974), which allow for cubic-time exhaustive shift-reduce parsing with context-free grammars (Billot and Lang, 1989).

Our work advances this line of research in two aspects. First, ours is more general than GLR in that it is not restricted to LR (a special case of shift-reduce), and thus does not require building an LR table, which is impractical for modern grammars with a large number of rules or features. In contrast, we employ the ideas behind GSS more flexibly to merge states based on features values, which can be viewed as constructing an implicit LR table *on-the-fly*. Second, unlike previous theoretical results about cubic-time complexity, we achieved linear-time performance by smart beam search with prefix cost inspired by Stolcke (1995), allowing for state-of-the-art data-driven parsing.

To the best of our knowledge, our work is the first linear-time incremental parser that performs dynamic programming. The parser of Roark and Hollingshead (2009) is also almost linear time, but they achieved this by discarding parts of the CKY chart, and thus do achieve incrementality.

6 Conclusion

We have presented a dynamic programming algorithm for shift-reduce parsing, which runs in linear-time in practice with beam search. This framework is general and applicable to a large-class of shift-reduce parsers, as long as the feature functions satisfy boundedness and monotonicity. Empirical results on a state-the-art dependency parser confirm the advantage of DP in many aspects: faster speed, larger search space, higher oracles, and better and faster learning. Our final parser outperforms all previously reported dependency parsers trained on the Penn Treebanks for both English and Chinese, and is much faster in speed (even with a Python implementation). For future work we plan to extend it to constituency parsing.

Acknowledgments

We thank David Chiang, Yoav Goldberg, Jonathan Graehl, Kevin Knight, and Roger Levy for helpful discussions and the three anonymous reviewers for comments. Mark-Jan Nederhof inspired the use of prefix cost. Yue Zhang helped with Chinese datasets, and Wenbin Jiang with feature sets. This work is supported in part by DARPA GALE Contract No. HR0011-06-C-0022 under subcontract to BBN Technologies, and by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

[9] Duan et al. (2007) and Zhang and Clark (2008) did not report word accuracies, but those can be recovered given non-root and root ones, and the number of non-punctuation words.

[10] Parser combination in Zhang and Clark (2008) achieves a higher word accuracy of 85.77%, but again, it is not directly comparable to our work.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey.

S. Billot and B. Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th ACL*, pages 143–151.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*, Ann Arbor, MI.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/PKDD*.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of ACL*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Lyn Frazier and Keith Rayner. 1982. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14(2):178 – 210.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT-2005)*.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*.

B. Lang. 1974. Deterministic techniques for efficient non-deterministic parsers. In *Automata, Languages and Programming, 2nd Colloquium*, volume 14 of *Lecture Notes in Computer Science*, pages 255–269, Saarbrücken. Springer-Verlag.

Lillian Lee. 2002. Fast context-free grammar parsing requires fast Boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, pages 135–143.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together. Workshop at ACL-2004*, Barcelona.

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

Brian Roark and Kristy Hollingshead. 2009. Linear complexity context-free parsing pipelines via chart constraints. In *Proceedings of HLT-NAACL*.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Masaru Tomita. 1988. Graph-structured stack and natural language parsing. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 249–257, Morristown, NJ, USA. Association for Computational Linguistics.

Masaru Tomita, editor. 1991. *Generalized LR Parsing*. Kluwer Academic Publishers.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*.

Hard Constraints for Grammatical Function Labelling

Wolfgang Seeker
University of Stuttgart
Institut für Maschinelle Sprachverarbeitung
seeker@ims.uni-stuttgart.de

Ines Rehbein
University of Saarland
Dep. for Comp. Linguistics & Phonetics
rehbein@coli.uni-sb.de

Jonas Kuhn
University of Stuttgart
Institut für Maschinelle Sprachverarbeitung
jonas@ims.uni-stuttgart.de

Josef van Genabith
Dublin City University
CNGL and School of Computing
josef@computing.dcu.ie

Abstract

For languages with (semi-) free word order (such as German), labelling grammatical functions on top of phrase-structural constituent analyses is crucial for making them interpretable. Unfortunately, most statistical classifiers consider only local information for function labelling and fail to capture important restrictions on the distribution of core argument functions such as subject, object etc., namely that there is at most one subject (etc.) per clause. We augment a statistical classifier with an integer linear program imposing hard linguistic constraints on the solution space output by the classifier, capturing global distributional restrictions. We show that this improves labelling quality, in particular for argument grammatical functions, in an intrinsic evaluation, and, importantly, grammar coverage for treebank-based (Lexical-Functional) grammar acquisition and parsing, in an extrinsic evaluation.

1 Introduction

Phrase or constituent structure is often regarded as an analysis step guiding semantic interpretation, while grammatical functions (i. e. subject, object, modifier etc.) provide important information relevant to determining predicate-argument structure.

In languages with restricted word order (e. g. English), core grammatical functions can often be recovered from configurational information in constituent structure analyses. By contrast, simple constituent structures are not sufficient for less configurational languages, which tend to encode grammatical functions by morphological means (Bresnan, 2001). Case features, for instance, can be important indicators of grammatical functions. Unfortunately, many of these languages (including German) exhibit strong syncretism where morphological cues can be highly ambiguous with respect to functional information.

Statistical classifiers have been successfully used to label constituent structure parser output with grammatical function information (Blaheta and Charniak, 2000; Chrupała and Van Genabith, 2006). However, as these approaches tend to use only limited and local context information for learning and prediction, they often fail to enforce simple yet important global linguistic constraints that exist for most languages, e. g. that there will be at most one subject (object) per sentence/clause.[1]

"Hard" linguistic constraints, such as these, tend to affect mostly the "core grammatical functions", i. e. the argument functions (rather than e. g. adjuncts) of a particular predicate. As these functions constitute the core meaning of a sentence (as in: who did what to whom), it is important to get them right. We present a system that adds grammatical function labels to constituent parser output for German in a postprocessing step. We combine a statistical classifier with an integer linear program (ILP) to model non-violable global linguistic constraints, restricting the solution space of the classifier to those labellings that comply with our set of global constraints. There are, of course, many other ways of including functional information into the output of a syntactic parser. Klein and Manning (2003) show that merging some linguistically motivated function labels with specific syntactic categories can improve the performance of a PCFG model on Penn-II En-

[1] Coordinate subjects/objects form a constituent that functions as a joint subject/object.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1087–1097,
Uppsala, Sweden, 11-16 July 2010.

glish data.[2] Tsarfaty and Sim'aan (2008) present a statistical model (Relational-Realizational Parsing) that alternates between functional and configurational information for constituency tree parsing and Hebrew data. Dependency parsers like the MST parser (McDonald and Pereira, 2006) and Malt parser (Nivre et al., 2007) use function labels as core part of their underlying formalism. In this paper, we focus on phrase structure parsing with function labelling as a post-processing step.

Integer linear programs have already been successfully used in related fields including semantic role labelling (Punyakanok et al., 2004), relation and entity classification (Roth and Yih, 2004), sentence compression (Clarke and Lapata, 2008) and dependency parsing (Martins et al., 2009). Early work on function labelling for German (Brants et al., 1997) reports 94.2% accuracy on gold data (a very early version of the TiGer Treebank (Brants et al., 2002)) using Markov models. Klenner (2007) uses a system similar to – but more restricted than – ours to label syntactic chunks derived from the TiGer Treebank. His research focusses on the correct selection of predefined subcategorisation frames for a verb (see also Klenner (2005)). By contrast, our research does not involve subcategorisation frames as an external resource, instead opting for a less knowledge-intensive approach. Klenner's system was evaluated on gold treebank data and used a small set of 7 dependency labels. We show that an ILP-based approach can be scaled to a large and comprehensive set of 42 labels, achieving 97.99% label accuracy on gold standard trees. Furthermore, we apply the system to automatically parsed data using a state-of-the-art statistical phrase-structure parser with a label accuracy of 94.10%. In both cases, the ILP-based approach improves the quality of argument function labelling when compared with a non-ILP-approach. Finally, we show that the approach substantially improves the quality and coverage (from 93.6% to 98.4%) of treebank-based Lexical-Functional Grammars for German over previous work in Rehbein and van Genabith (2009).

The paper is structured as follows: Section 2 presents basic data demonstrating the challenges presented by German word order and case syncretism for the function labeller. Section 3 describes the labeller including the feature model of the classifier and the integer linear program used to pick the correct labelling. The evaluation part (Section 4) is split into an intrinsic evaluation measuring the quality of the labelling directly using the German TiGer Treebank (Brants et al., 2002), and an extrinsic evaluation where we test the impact of the constraint-based labelling on treebank-based automatic LFG grammar acquisition.

2 Data

Unlike English, German exhibits a relatively free word order, i. e. in main clauses, the verb occupies second position (the last position in subordinated clauses) and arguments and adjuncts can be placed (fairly) freely. The grammatical function of a noun phrase is marked morphologically on its constituting parts. Determiners, pronouns, adjectives and nouns carry case markings and in order to be well-formed, all parts of a noun phrase have to agree on their case features. German uses a nominative–accusative system to mark predicate arguments. Subjects are marked with nominative case, direct objects carry accusative case. Furthermore, indirect objects are mostly marked with dative case and sometimes genitive case.

(1) *Der Löwe* *gibt* *dem Wolf* *einen Besen.*
NOM DAT ACC
the lion gives the wolf a broom
The lion gives a broom to the wolf.

(1) shows a sentence containing the ditransitive verb *geben* (to give) with its three arguments. Here, the subject is unambiguously marked with nominative case (NOM), the indirect object with dative case (DAT) and the direct object with accusative case (ACC). (2) shows possible word orders for the arguments in this sentence.[3]

(2) Der Löwe gibt einen Besen dem Wolf.
Dem Wolf gibt der Löwe einen Besen.
Dem Wolf gibt einen Besen der Löwe.
Einen Besen gibt der Löwe dem Wolf.
Einen Besen gibt dem Wolf der Löwe.

Since all permutations of arguments are possible, there is no chance for a statistical classifier to decide on the correct function of a noun phrase by its position alone. Introducing adjuncts to this example makes matters even worse.

[2] Table 6 shows that for our data a model with merged category and function labels (but without hard constraints!) performs slightly worse than the ILP approach developed in this paper.

[3] Note that although (apart from the position of the finite verb) there are no syntactic restrictions on the word order, there are restrictions pertaining to phonological or information structure.

Case information for a given noun phrase can give a classifier some clue about the correct argument function, since functions are strongly related to case values. Unfortunately, the German case system is complex (see Eisenberg (2006) for a thorough description) and exhibits a high degree of case syncretism. (3) shows a sentence where both argument NPs are ambiguous between nominative or accusative case. In such cases, additional semantic or contextual information is required for disambiguation. A statistical classifier (with access to local information only) runs a high risk of incorrectly classifying both NPs as subjects, or both as direct objects or even as nominal predicates (which are also required to carry nominative case). This would leave us with uninterpretable results. Uninterpretability of this kind can be avoided if we are able to constrain the number of subjects and objects globally to one per clause.[4]

(3) *Das Schaf* *sieht* *das Mädchen.*
NOM/ACC NOM/ACC
the sheep sees the girl
EITHER The sheep sees the girl
OR The girl sees the sheep.

3 Grammatical Function Labelling

Our function labeller was developed and tested on the TiGer Treebank (Brants et al., 2002). The TiGer Treebank is a phrase-structure and grammatical function annotated treebank with 50,000 newspaper sentences from the *Frankfurter Rundschau* (Release 2, July 2006). Its overall annotation scheme is quite flat to account for the relatively free word order of German and does not allow for unary branching. The annotations use non-projective trees modelling long distance dependencies directly by crossing branches. Words are lemmatised and part-of-speech tagged with the *Stuttgart-Tübingen Tag Set (STTS)* (Schiller et al., 1999) and contain morphological annotations (Release 2). TiGer uses 25 syntactic categories and a set of 42 function labels to annotate the grammatical function of a phrase.

The function labeller consists of two main components, a maximum entropy classifier and an integer linear program. This basic architecture was introduced by Punyakanok et al. (2004) for the task of semantic role labelling and since then has been applied to different NLP tasks without significant changes. In our case, its input is a bare tree structure (as obtained by a standard phrase structure parser) and it outputs a tree structure where every node is labelled with the grammatical relation it bears to its mother node. For each possible label and for each node, the classifier assigns a probability that this node is labelled by this label. This results in a complete probability distribution over all labels for each node. An integer linear program then tries to find the optimal overall tree labelling by picking for each node the label with the highest probability without violating any of its constraints. These constraints implement linguistic rules like the *one-subject-per-sentence* rule mentioned above. They can also be used to capture treebank particulars, such as for example that punctuation marks never receive a label.

3.1 The Feature Model

Maximum entropy classifiers have been used in a wide range of applications in NLP for a long time (Berger et al., 1996; Ratnaparkhi, 1998). They usually give good results while at the same time allowing for the inclusion of arbitrarily complex features. They also have the advantage that they directly output probability distributions over their set of labels (unlike e. g. SVMs).

The classifier uses the following features:

- the lemma (if terminal node)
- the category (the POS for terminal nodes)
- the number of left/right sisters
- the category of the two left/right sisters
- the number of daughters
- the number of terminals covered
- the lemma of the left/right corner terminal
- the category of the left/right corner terminal
- the category of the mother node
- the category of the mother's head node
- the lemma of the mother's head node
- the category of the grandmother node
- the category of the grandmother's head node
- the lemma of the grandmother's head node
- the case features for noun phrases
- the category for PP objects
- the lemma for PP objects (if terminal node)

These features are also computed for the head of the phrase, determined using a set of head-finding rules in the style of Magerman (1995) adapted to TiGer. For lemmatisation, we use TreeTagger (Schmid, 1994) and case features of noun

[4] Although the classifier may, of course, still identify the wrong phrase as subject or object.

phrases are obtained from a full German morphological analyser based on (Schiller, 1994). If a noun phrase consists of a single word (e. g. pronouns, but also bare common nouns and proper nouns), all case values output by the analyser are used to reflect the case syncretism. For multi-word noun phrases, the case feature is computed by taking the intersection of all case-bearing words inside the noun phrase, i. e. determiners, pronouns, adjectives, common nouns and proper nouns. If, for some reason (e.g., due to a bracketing error in phrase structure parsing), the intersection turns out to be empty, all four case values are assigned to the phrase.[5]

3.2 Constrained Optimisation

In the second step, a binary integer linear program is used to select those labels that optimise the whole tree labelling. A linear program consists of a linear objective function that is to be maximised (or minimised) and a set of constraints which impose conditions on the variables of the objective function (see (Clarke and Lapata, 2008) for a short but readable introduction). Although solving a linear program has polynomial complexity, requiring the variables to be integral or binary makes finding a solution exponentially hard in the worst case. Fortunately, there are efficient algorithms which are capable of handling a large number of variables and constraints in practical applications.[6]

For the function labeller, we define the set of binary variables $V = N \times L$ to be the crossproduct of the set of nodes N and the set of labels L. Setting a variable $x_{n,l}$ to 1 means that node n is labelled by label l. Every variable is weighted by the probability $w_{n,l} = P(l|f(n))$ which the classifier has assigned to this node-label combination. The objective function that we seek to optimise is defined as the sum over all weighted variables:

$$\max \sum_{n \in N} \sum_{l \in L} w_{n,l} x_{n,l} \tag{4}$$

Since we want every node to receive exactly one label, we add a constraint that for every node n, exactly one of its variables is set to 1.

$$\sum_{l \in L} x_{n,l} = 1 \tag{5}$$

Up to now, the whole system is doing exactly the same as an ordinary classifier that always takes the most probable label for each node. We will now add additional global and local linguistic constraints.[7]

The first and most important constraint restricts the number of each argument function (as opposed to modifier functions) to at most one per clause. Let $D \subset N \times N$ be the direct dominance relation between the nodes of the current tree. For every node n with category S (sentence) or VP (verb phrase), at most one of its daughters is allowed to be labelled SB (subject). The single-subject-function condition is defined as:

$$cat(n) \in \{S, VP\} \longrightarrow \sum_{\langle n,m \rangle \in D} x_{m,SB} \leq 1 \tag{6}$$

Identical constraints are added for labels OA, OA2, DA, OG, OP, PD, OC, EP.[8]

We add further constraints to capture the following linguistic restrictions:

- Of all daughters of a phrase, only one is allowed to be labelled HD (head).

$$\sum_{\langle n,m \rangle \in D} x_{m,HD} \leq 1 \tag{7}$$

- If a noun phrase carries no case feature for nominative case, it cannot be labelled SB, PD or EP.

$$case(n) \neq nom \longrightarrow \sum_{l \in \{SB,PD,EP\}} x_{n,l} = 0 \tag{8}$$

- If a noun phrase carries no case feature for accusative case, it cannot be labelled OA or OA2.
- If a noun phrase carries no case feature for dative case, it cannot be labelled DA.
- If a noun phrase carries no case feature for genitive case, it cannot be labelled OG or AG[9].

[5] We decided to train the classifier on automatically assigned and possibly ambiguous morphological information instead of on the hand-annotated and manually disambiguated morphological information provided by TiGer because we want the classifier to learn the German case syncretism. This way, the classifier will perform better when presented with unseen data (e.g. from parser output) for which no hand-annotated morphological information is available.

[6] See lpsolve (http://lpsolve.sourceforge.net/) or GLPK (http://www.gnu.org/software/glpk/glpk.html) for open-source implementations

[7] Note that some of these constraints are language specific in that they represent linguistic facts about German and do not necessarily hold for other languages. Furthermore, the constraints are treebank specific to a certain degree in that they use a TiGer-specific set of labels and are conditioned on TiGer-specific configurations and categories.

[8] SB = subject, OA = accusative object, OA2 = second accusative object, DA = dative, OG = genitive object, OP = prepositional object, PD = predicate, OC = clausal object, EP = expletive es

[9] AG = genitive adjunct

Unlike Klenner (2007), we do not use predefined subcategorization frames, instead letting the statistical model choose arguments.

In TiGer, sentences whose main verbs are formed from auxiliary-participle combinations, are annotated by embedding the participle under an extra VP node and non-subject arguments are sisters to the participle. Therefore we add an extension of the constraint in (6) to the constraint set in order to also include the daughters of an embedded VP node in such a case.

Because of the particulars of the annotation scheme of TiGer, we can decide some labels in advance. As mentioned before, punctuation does not get a label in TiGer. We set the label for those nodes to $--$ (no label). Other examples are:

- If a node's category is PTKVZ (separated verb particle), it is labeled SVP (separable verb particle).

$$cat(n) = PTKVZ \longrightarrow x_{n,SVP} = 1 \quad (9)$$

- If a node's category is APPR, APPRART, APPO or APZR (prepositions), it is labeled AC (adpositional case marker).
- All daughters of an MTA node (multi-token adjective) are labeled ADC (adjective component).

These constraints are conditioned on part-of-speech tags and require high POS-tagging accuracy (when dealing with raw text).

Due to the constraints imposed on the classification, the function labeller can no longer assign two subjects to the same S node. Faced with two nodes whose most probable label is SB, it has to decide on one of them taking the next best label for the other. This way, it outputs the optimal solution with respect to the set of constraints. Note that this requires the feature model not only to rank the correct label highest but also to provide a reasonable ranking of the other labels as well.

4 Evaluation

We conducted a number of experiments using 1,866 sentences of the TiGer Dependency Bank (Forst et al., 2004) as our test set. The TiGerDB is a part of the TiGer Treebank semi-automatically converted into a dependency representation. We use the manually labelled TiGer trees corresponding to the sentences in the TiGerDB for assessing the labelling quality in the intrinsic evaluation, and the dependencies from TiGerDB for assessing the quality and coverage of the automatically acquired LFG resources in the extrinsic evaluation.

In order to test on real parser output, the test set was parsed with the Berkeley Parser (Petrov et al., 2006) trained on 48k sentences of the TiGer corpus (Table 1), excluding the test set. Since the Berkeley Parser assumes projective structures, the training data and test data were made projective by raising non-projective nodes in the tree (Kübler, 2005).

precision	83.60	recall	82.81
f-score	83.20	tagging acc.	97.97

Table 1: evalb unlabelled parsing scores on test set for Berkeley Parser trained on 48,000 sentences (sentence length ≤ 40)

The maximum entropy classifier of the function labeller was trained on 46,473 sentences of the TiGer Treebank (excluding the test set) which yields about 1.2 million nodes as training samples. For training the Maximum Entropy Model, we used the BLMVM algorithm (Benson and More, 2001) with a width factor of 1.0 (Kazama and Tsujii, 2005) implemented in an open-source C++ library from Tsujii Laboratory.[10] The integer linear program was solved with the simplex algorithm in combination with a branch-and-bound method using the freely available GLPK.[11]

4.1 Intrinsic Evaluation

In the intrinsic evaluation, we measured the quality of the labelling itself. We used the node span evaluation method of (Blaheta and Charniak, 2000) which takes only those nodes into account which have been recognised correctly by the parser, i.e. if there are two nodes in the parse and the reference treebank tree which cover the same word span. Unlike Blaheta and Charniak (2000) however, we do not require the two nodes to carry the same syntactic category label.[12]

Table 2 shows the results of the node span evaluation. The labeller achieves close to 98% label accuracy on gold treebank trees which shows that the feature model captures the differences between the individual labels well. Results on parser output are about 4 percentage points (absolute) lower as parsing errors can distort local context features for the classifier even if the node itself has been parsed

[10] http://www-tsujii.is.s.u-tokyo.ac.jp/∼tsuruoka/maxent/

[11] http://www.gnu.org/software/glpk/glpk.html

[12] We also excluded the root node, all punctuation marks and both nodes in unary branching sub-trees from evaluation.

correctly. The addition of the ILP constraints improves results only slightly since the constraints affect only (a small number of) argument labels while the evaluation considers all 40 labels occurring in the test set. Since the constraints restrict the selection of certain labels, a less probable label has to be picked by the labeller if the most probable is not available. If the classifier is ranking labels sensibly, the correct label should emerge. However, with an incorrect ranking, the ILP constraints might also introduce new errors.

	label accuracy	error red.
without constraints		
gold	44689/45691 = 97.81%	–
parser	40578/43140 = 94.06%	–
with constraints		
gold	44773/45691 = 97.99%*	8.21%
parser	40593/43140 = 94.10%	0.68%

Table 2: label accuracy and error reduction (all labels) for node span evaluation, * statistically significant, sign test, α = 0.01 (Koo and Collins, 2005)

As the main target of the constraint set are argument functions, we also tested the quality of argument labels. Table 3 shows the node span evaluation in terms of precision, recall and f-score for argument functions only, with clear statistically significant improvements.

	prec.	rec.	f-score
without constraints			
gold standard	92.41	91.86	92.13
parser output	88.14	86.43	87.28
with constraints			
gold standard	94.31	92.76	93.53*
parser output	89.51	86.73	88.09*

Table 3: node span results for the test set, argument functions only (SB, EP, PD, OA, OA2, DA, OG, OP, OC), * statistically significant, sign test, α = 0.01 (Koo and Collins, 2005)

For comparison and to establish a highly competitive baseline, we use the best-scoring system in (Chrupała and Van Genabith, 2006), trained and tested on exactly the same data sets. This purely statistical labeller achieves accuracy of 96.44% (gold) and 92.81% (parser) for all labels, and f-scores of 89.88% (gold) and 84.98% (parser) for argument labels. Tables 2 and 3 show that our system (with and even without ILP constraints) comprehensively outperforms all corresponding baseline scores.

The node span evaluation defines a correct labelling by taking only those nodes (in parser output) into account that have a corresponding node in the reference tree. However, as this restricts attention to correctly parsed nodes, the results are somewhat over-optimistic. Table 4 provides the results obtained from an evalb evaluation of the same data sets.[13] The gold standard scores are high confirming our previous findings about the performance of the function labeller. However, the results on parser output are much worse. The evaluation scores are now taking the parsing quality into account (Table 1). The considerable drop in quality between gold trees and parser output clearly shows that a good parse tree is an important prerequisite for reasonable function labelling. This is in accordance with previous findings by Punyakanok et al. (2008) who emphasise the importance of syntactic parsing for the closely related task of semantic role labelling.

	prec.	rec.	f-score
without constraints			
gold standard	95.94	95.94	95.94
parser output	76.27	75.55	75.91
with constraints			
gold standard	96.21	96.21	96.21
parser output	76.36	75.64	76.00

Table 4: evalb results for the test set

4.1.1 Subcategorisation Frames

Early on in the paper we mention that, unlike e. g. Klenner (2007), we did not include predefined subcategorisation frames into the constraint set, but rather let the joint statistical and ILP models decide on the correct type of arguments assigned to a verb. The assumption is that if one uses predefined subcategorisation frames which fix the number and type of arguments for a verb, one runs the risk of excluding correct labellings due to missing subcat frames, unless a very comprehensive and high quality subcat lexicon resource is available.

In order to test this assumption, we run an additional experiment with about 10,000 verb frames for 4,508 verbs, which were automatically extracted from our training section. Following Klenner (2007), for each verb and for each subcat frame for this verb attested at least once in the training data, we introduce a new binary variable f_n to the ILP model representing the n-th frame (for the verb) weighted by its frequency.

We add an ILP constraint requiring exactly one of the frames to be set to one (each verb has to have a subcat frame) and replace the ILP constraint in (6) by:

[13] Function labels were merged with the category symbols.

$$\sum_{\langle n,m\rangle \in D} x_{m,SB} - \sum_{SB \in f_i} f_i = 0 \qquad (10)$$

This constraint requires the number of subjects in a phrase to be equal to the number of selected[14] verb frames that require a subject. As each verb is constrained to "select" exactly one subcat frame (see additional ILP constraint above), there is at most one subject per phrase, if the frame in question requires a subject. If the selected frame does not require a subject, then the constraint blocks the assignment of subjects for the entire phrase. The same was done for the other argument functions and as before we included an extension of this constraint to cover embedded VPs. For unseen verbs (i.e. verbs not attested in the training set) we keep the original constraints as a back-off.

	prec.	rec.	f-score
all labels (cmp. Table 2)			
gold standard	97.24	97.24	97.24
parser output	93.43	93.43	93.43
argument functions only (cmp. Table 3)			
gold standard	91.36	90.12	90.74
parser output	86.64	84.38	85.49

Table 5: node span results for the test set using constraints with automatically extracted subcat frames

Table 5 shows the results of the test set node span evaluation when using the ILP system enhanced with subcat frames. Compared to Tables 2 and 3, the results are clearly inferior, and particularly so for argument grammatical functions. This seems to confirm our assumption that, given our data, letting the joint statistical and ILP model decide argument functions is superior to an approach that involves subcat frames. However, and importantly, our results do not rule out that a more comprehensive subcat frame resource may in fact result in improvements.

4.2 Extrinsic Evaluation

Over the last number of years, treebank-based deep grammar acquisition has emerged as an attractive alternative to hand-crafting resources within the HPSG, CCG and LFG paradigms (Miyao et al., 2003; Clark and Hockenmaier, 2002; Cahill et al., 2004). While most of the initial development work focussed on English, more recently efforts have branched to other languages. Below we concentrate on LFG.

[14]The variable representing this frame has been set to 1.

Lexical-Functional Grammar (Bresnan, 2001) is a constraint-based theory of grammar with minimally two levels of representation: c(onstituent)-structure and f(unctional)-structure. C-structure (CFG trees) captures language specific surface configurations such as word order and the hierarchical grouping of words into phrases, while f-structure represents more abstract (and somewhat more language independent) grammatical relations (essentially bilexical labelled dependencies with some morphological and semantic information, approximating to basic predicate-argument structures) in the form of attribute-value structures. F-structures are defined in terms of equations annotated to nodes in c-structure trees (grammar rules). Treebank-based LFG acquisition was originally developed for English (Cahill, 2004; Cahill et al., 2008) and is based on an f-structure annotation algorithm that annotates c-structure trees (from a treebank or parser output) with f-structure equations, which are read off of the tree and passed on to a constraint solver producing an f-structure for the given sentence. The English annotation algorithm (for Penn-II treebank-style trees) relies heavily on configurational and categorial information, translating this into grammatical functional information (subject, object etc.) represented at f-structure. LFG is "functional" in the mathematical sense, in that argument grammatical functions have to be single valued (there cannot be two or more subjects etc. in the same clause). In fact, if two or more values are assigned to a single argument grammatical function in a local tree, the LFG constraint solver will produce a clash (i. e. it will fail to produce an f-structure) and the sentence will be considered ungrammatical (in other words, the corresponding c-structure tree will be uninterpretable).

Rehbein (2009) and Rehbein and van Genabith (2009) develop an f-structure annotation algorithm for German based on the TiGer treebank resource. Unlike the English annotation algorithm and because of the language-particular properties of German (see Section 2), the German annotation algorithm cannot rely on c-structure configurational information, but instead heavily uses TiGer function labels in the treebank. Learning function labels is therefore crucial to the German LFG annotation algorithm, in particular when parsing raw text. Because of the strong case syncretism in German, traditional classification models using local

information only run the risk of predicting multiple occurences of the same function (subject, object etc.) at the same level, causing feature clashes in the constraint solver with no f-structure being produced. Rehbein (2009) and Rehbein and van Genabith (2009) identify this as a major problem resulting in a considerable loss in coverage of the German annotation algorithm compared to English, in particular for parsing raw text, where TiGer function labels have to be supplied by a machine-learning-based method and where the coverage of the LFG annotation algorithm drops to 93.62% with corresponding drops in recall and f-scores for the f-structure evaluations (Table 6).

Below we test whether the coverage problems caused by incorrect multiple assignments of grammatical functions can be addressed using the combination of classifier with ILP constraints developed in this paper. We report experiments where automatically parsed and labelled data are handed over to an LFG f-structure computation algorithm. The f-structures produced are converted into a dependency triple representation (Crouch et al., 2002) and evaluated against TiGerDB.

	cov.	prec.	rec.	f-score
upper bound	99.14	85.63	82.58	84.07
without constraints				
gold	95.82	84.71	76.68	80.49
parser	93.41	79.70	70.38	74.75
with constraints				
gold	99.30	84.62	82.15	83.37
parser	98.39	79.43	75.60	77.47
Rehbein 2009				
parser	93.62	79.20	68.86	73.67

Table 6: f-structure evaluation results for the test set against TigerDB

Table 6 shows the results of the f-structure evaluation against TiGerDB, with 84.07% f-score upper-bound results for the f-structure annotation algorithm on the original TiGer treebank trees with hand-annotated function labels. Using the function labeller without ILP constraints results in drastic drops in coverage (between 4.5% and 6.5% points absolute) and hence recall (6% and 12%) and f-score (3.5% and 9.5%) for both gold trees and parser output (compared to upper bounds). By contrast, with ILP constraints, the loss in coverage observed above almost completely disappears and recall and f-scores improve by between 4.4% and 5.5% (recall) and 3% (f-score) absolute (over without ILP constraints). For comparison, we repeated the experiment using the best-scoring method of Rehbein (2009). Rehbein trains the Berkeley Parser to learn an extended category set, merging TiGer function labels with syntactic categories, where the parser outputs fully-labelled trees. The results show that this approach suffers from the same drop in coverage as the classifier without ILP constraints, with recall about 7% and f-score about 4% (absolute) lower than for the classifier with ILP constraints.

Table 7 shows the dramatic effect of the ILP constraints on the number of sentences in the test set that have multiple argument functions of the same type within the same clause. With ILP constraints, the problem disappears and therefore, less feature-clashes occur during f-structure computation.

	no constraints	constraints
gold	185	0
parser	212	0

Table 7: Number of sentences in the test set with doubly annotated argument functions

In order to assess whether ILP constraints help with coverage only or whether they affect the quality of the f-structures as well, we repeat the experiment in Table 6, however this time evaluating only on those sentences that receive an f-structure, ignoring the rest. Table 8 shows that the impact of ILP constraints on quality is much less dramatic than on coverage, with only very small variations in precison, recall and f-scores across the board, and small increases over Rehbein (2009).

	cov.	prec.	rec.	f-score
no constr.	93.41	79.70	77.89	78.79
constraints	98.39	79.43	77.85	78.64
Rehbein	93.62	79.20	76.43	77.79

Table 8: f-structure evaluation results for parser output excluding sentences without f-structures

Early work on automatic LFG acquisition and parsing for German is presented in Cahill et al. (2003) and Cahill (2004), adapting the English Annotation Algorithm to an earlier and smaller version of the TiGer treebank (without morphological information) and training a parser to learn merged Tiger function-category labels, and reporting 95.75% coverage and an f-score of 74.56% f-structure quality against 2,000 gold treebank trees automatically converted into f-structures. Rehbein (2009) uses the larger Release 2 of the treebank (with morphological information) reporting 77.79% f-score and coverage of 93.62% (Ta-

ble 8) against the dependencies in the TiGerDB test set. The only rule-based approach to German LFG-parsing we are aware of is the hand-crafted German grammar in the ParGram Project (Butt et al., 2002). Forst (2007) reports 83.01% dependency f-score evaluated against a set of 1,497 sentences of the TiGerDB. It is very difficult to compare results across the board, as individual papers use (i) different versions of the treebank, (ii) different (sections of) gold-standards to evaluate against (gold TiGer trees in TigerDB, the dependency representations provided by TigerDB, automatically generated gold-standards etc.) and (iii) different label/grammatical function sets. Furthermore, (iv) coverage differs drastically (with the hand-crafted LFG resources achieving about 80% full f-structures) and finally, (v) some of the grammars evaluated having been used in the generation of the gold standards, possibly introducing a bias towards these resources: the German hand-crafted LFG was used to produce TiGerDB (Forst et al., 2004). In order to put the results into some perspective, Table 9 shows an evaluation of our resources against a set of automatically generated gold standard f-structures produced by using the f-structure annotation algorithm on the original hand-labelled TiGer gold trees in the section corresponding to TiGerDB: without ILP constraints we achieve a dependency f-score of 84.35%, with ILP constraints 87.23% and 98.89% coverage.

	cov.	prec.	rec.	f-score
without constraints				
gold	95.24	97.76	90.93	94.22
parser	93.35	88.71	80.40	84.35
with constraints				
gold	99.30	97.66	97.33	97.50
parser	98.89	88.37	86.12	87.23

Table 9: f-structure evaluation results for the test set against automatically generated goldstandard (1,850 sentences)

5 Conclusion

In this paper, we addressed the problem of assigning grammatical functions to constituent structures. We have proposed an approach to grammatical function labelling that combines the flexibility of a statistical classifier with linguistic expert knowledge in the form of hard constraints implemented by an integer linear program. These constraints restrict the solution space of the classifier by blocking those solutions that cannot be correct. One of the strengths of an integer linear program is the unlimited context it can take into account by optimising over the entire structure, providing an elegant way of supporting classifiers with explicit linguistic knowledge while at the same time keeping feature models small and comprehensible. Most of the constraints are direct formalizations of linguistic generalizations for German. Our approach should generalise to other languages for which linguistic expertise is available.

We evaluated our system on the TiGer corpus and the TiGerDB and gave results on gold standard trees and parser output. We also applied the German f-structure annotation algorithm to the automatically labelled data and evaluated the system by measuring the quality of the resulting f-structures. We found that by using the constraint set, the function labeller ensures the interpretability and thus the usefulness of the syntactic structure for a subsequently applied processing step. In our f-structure evaluation, that means, the f-structure computation algorithm is able to produce an f-structure for almost all sentences.

Acknowledgements

The first author would like to thank Gerlof Bouma for a lot of very helpful discussions. We would like to thank our anonymous reviewers for detailed and helpful comments. The research was supported by the Science Foundation Ireland SFI (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) and by DFG (German Research Foundation) through SFB 632 Potsdam-Berlin and SFB 732 Stuttgart.

References

Steven J. Benson and Jorge J. More. 2001. A limited memory variable metric method in subspaces and bound constrained optimization problems. Technical report, Argonne National Laboratory.

Adam L. Berger, Vincent J.D. Pietra, and Stephen A.D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):71.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 234 – 240, Seattle, Washington. Morgan Kaufmann Publishers Inc.

Thorsten Brants, Wojciech Skut, and Brigitte Krenn. 1997. Tagging grammatical functions. In *Proceedings of EMNLP*, volume 97, pages 64–74.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, page 2441.

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell Publishers.

Miriam Butt, Helge Dyvik, Tracy Halloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *COLING-02 on Grammar engineering and evaluation-Volume 15*, volume pages, page 7. Association for Computational Linguistics.

Aoife Cahill, Martin Forst, Mairead McCarthy, Ruth ODonovan, Christian Rohrer, Josef van Genabith, and Andy Way. 2003. Treebank-based multilingual unification-grammar development. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development at the 15th ESSLLI*, page 1724.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 319–es.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. 2008. Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation. *Computational Linguistics*, 34(1):81–124, März.

Aoife Cahill. 2004. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. Ph.D. thesis, Dublin City University.

Grzegorz Chrupała and Josef Van Genabith. 2006. Using machine-learning to assign function labels to parser output for Spanish. In *Proceedings of the COLING/ACL main conference poster session*, page 136143, Sydney. Association for Computational Linguistics.

Stephen Clark and Judith Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proceedings of the LREC 2002*, pages 60–66.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Richard Crouch, Ronald M. Kaplan, Tracy Halloway King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad-coverage stochastic parser. In *Proceedings of LREC 2002 Workshop*, pages 67–74, Las Palmas, Canary Islands, Spain.

Peter Eisenberg. 2006. *Grundriss der deutschen Grammatik: Das Wort*. J.B. Metzler, Stuttgart, 3 edition.

Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Shirra, and Valia Kordoni. 2004. Towards a dependency-based gold standard for German parsers The TiGer Dependency Bank. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora (LINC '04)*, Geneva, Switzerland.

Martin Forst. 2007. Filling Statistics with Linguistics Property Design for the Disambiguation of German LFG Parses. In *Proceedings of ACL 2007*. Association for Computational Linguistics.

Jun'Ichi Kazama and Jun'Ichi Tsujii. 2005. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1):159194.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL 2003*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

Manfred Klenner. 2005. Extracting Predicate Structures from Parse Trees. In *Proceedings of the RANLP 2005*.

Manfred Klenner. 2007. Shallow dependency labeling. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, page 201204, Prague. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 507–514, Morristown, NJ, USA. Association for Computational Linguistics.

Sandra Kübler. 2005. How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples And Oranges. In *Proceedings of RANLP 2005*, Borovets, Bulgaria.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, page 276283, Morristown, NJ, USA. Association for Computational Linguistics Morristown, NJ, USA.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL 2009*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the Conference on Recent Advances in Natural Language Processing RANLP 2003*, volume 2.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, Januar.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL - ACL '06*, pages 433–440, Morristown, NJ, USA. Association for Computational Linguistics.

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

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257–287, Juni.

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

Ines Rehbein and Josef van Genabith. 2009. Automatic Acquisition of LFG Resources for German-As Good as it gets. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of LFG Conference 2009*. CSLI Publications.

Ines Rehbein. 2009. *Treebank-based grammar acquisition for German*. Ph.D. thesis, Dublin City University.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNNL 2004*.

Anne Schiller, Simone Teufel, and Christine Stöckert. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset). Technical Report August, Universität Stuttgart.

Anne Schiller. 1994. Dmor - user's guide. Technical report, University of Stuttgart.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12. Manchester, UK.

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08*, pages 889–896, Morristown, NJ, USA. Association for Computational Linguistics.

Simple, Accurate Parsing with an All-Fragments Grammar

Mohit Bansal and **Dan Klein**
Computer Science Division
University of California, Berkeley
{mbansal, klein}@cs.berkeley.edu

Abstract

We present a simple but accurate parser which exploits both large tree fragments and symbol refinement. We parse with *all* fragments of the training set, in contrast to much recent work on tree selection in data-oriented parsing and tree-substitution grammar learning. We require only simple, deterministic grammar symbol refinement, in contrast to recent work on latent symbol refinement. Moreover, our parser requires no explicit lexicon machinery, instead parsing input sentences as character streams. Despite its simplicity, our parser achieves accuracies of over 88% F1 on the standard English WSJ task, which is competitive with substantially more complicated state-of-the-art lexicalized and latent-variable parsers. Additional specific contributions center on making implicit all-fragments parsing efficient, including a coarse-to-fine inference scheme and a new graph encoding.

1 Introduction

Modern NLP systems have increasingly used data-intensive models that capture many or even all substructures from the training data. In the domain of syntactic parsing, the idea that all training fragments[1] might be relevant to parsing has a long history, including tree-substitution grammar (data-oriented parsing) approaches (Scha, 1990; Bod, 1993; Goodman, 1996a; Chiang, 2003) and tree kernel approaches (Collins and Duffy, 2002). For machine translation, the key modern advancement has been the ability to represent and memorize large training substructures, be it in contiguous phrases (Koehn et al., 2003) or syntactic trees (Galley et al., 2004; Chiang, 2005; Deneefe and Knight, 2009). In all such systems, a central challenge is efficiency: there are generally a combinatorial number of substructures in the training data, and it is impractical to explicitly extract them all. On both efficiency and statistical grounds, much recent TSG work has focused on fragment selection (Zuidema, 2007; Cohn et al., 2009; Post and Gildea, 2009).

At the same time, many high-performance parsers have focused on *symbol refinement* approaches, wherein PCFG independence assumptions are weakened not by increasing rule sizes but by subdividing coarse treebank symbols into many subcategories either using structural annotation (Johnson, 1998; Klein and Manning, 2003) or lexicalization (Collins, 1999; Charniak, 2000). Indeed, a recent trend has shown high accuracies from models which are dedicated to inducing such subcategories (Henderson, 2004; Matsuzaki et al., 2005; Petrov et al., 2006). In this paper, we present a simplified parser which combines the two basic ideas, using both large fragments and symbol refinement, to provide non-local and local context respectively. The two approaches turn out to be highly complementary; even the simplest (deterministic) symbol refinement and a basic use of an all-fragments grammar combine to give accuracies substantially above recent work on tree-substitution grammar based parsers and approaching top refinement-based parsers. For example, our best result on the English WSJ task is an F1 of over 88%, where recent TSG parsers[2] achieve 82-84% and top refinement-based parsers[3] achieve 88-90% (e.g., Table 5).

Rather than select fragments, we use a simplification of the PCFG-reduction of DOP (Goodman,

[1] In this paper, a *fragment* means an elementary tree in a tree-substitution grammar, while a *subtree* means a fragment that bottoms out in terminals.

[2] Zuidema (2007), Cohn et al. (2009), Post and Gildea (2009). Zuidema (2007) incorporates deterministic refinements inspired by Klein and Manning (2003).

[3] Including Collins (1999), Charniak and Johnson (2005), Petrov and Klein (2007).

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1098–1107,
Uppsala, Sweden, 11-16 July 2010.

1996a) to work with all fragments. This reduction is a flexible, implicit representation of the fragments that, rather than extracting an intractably large grammar over fragment *types*, indexes all nodes in the training treebank and uses a compact grammar over indexed node *tokens*. This indexed grammar, when appropriately marginalized, is equivalent to one in which all fragments are explicitly extracted. Our work is the first to apply this reduction to *full-scale* parsing. In this direction, we present a coarse-to-fine inference scheme and a compact graph encoding of the training set, which, together, make parsing manageable. This tractability allows us to avoid selection of fragments, and work with all fragments.

Of course, having a grammar that includes all training substructures is only desirable to the extent that those structures can be appropriately weighted. Implicit representations like those used here do not allow arbitrary weightings of fragments. However, we use a simple weighting scheme which does decompose appropriately over the implicit encoding, and which is flexible enough to allow weights to depend not only on frequency but also on fragment size, node patterns, and certain lexical properties. Similar ideas have been explored in Bod (2001), Collins and Duffy (2002), and Goodman (2003). Our model empirically affirms the effectiveness of such a flexible weighting scheme in full-scale experiments.

We also investigate parsing without an explicit lexicon. The all-fragments approach has the advantage that parsing down to the character level requires no special treatment; we show that an explicit lexicon is not needed when sentences are considered as strings of characters rather than words. This avoids the need for complex unknown word models and other specialized lexical resources.

The main contribution of this work is to show practical, tractable methods for working with an all-fragments model, without an explicit lexicon. In the parsing case, the central result is that accuracies in the range of state-of-the-art parsers (i.e., over 88% F1 on English WSJ) can be obtained with no sampling, no latent-variable modeling, no smoothing, and even no explicit lexicon (hence negligible training overall). These techniques, however, are not limited to the case of monolingual parsing, offering extensions to models of machine translation, semantic interpretation, and other areas in which a similar tension exists between the desire to extract many large structures and the computational cost of doing so.

2 Representation of Implicit Grammars

2.1 All-Fragments Grammars

We consider an *all-fragments grammar* G (see Figure 1(a)) derived from a binarized treebank B. G is formally a tree-substitution grammar (Resnik, 1992; Bod, 1993) wherein each subgraph of each training tree in B is an elementary tree, or *fragment* f, in G. In G, each derivation d is a tree (multiset) of fragments (Figure 1(c)), and the weight of the derivation is the product of the weights of the fragments: $\omega(d) = \prod_{f \in d} \omega(f)$. In the following, the derivation weights, when normalized over a given sentence s, are interpretable as conditional probabilities, so G induces distributions of the form $P(d|s)$.

In models like G, many derivations will generally correspond to the same unsegmented tree, and the parsing task is to find the tree whose sum of derivation weights is highest: $t_{max} = \arg\max_t \sum_{d \in t} \omega(d)$. This final optimization is intractable in a way that is orthogonal to this paper (Sima'an, 1996); we describe minimum Bayes risk approximations in Section 4.

2.2 Implicit Representation of G

Explicitly extracting all fragment-rules of a grammar G is memory and space intensive, and impractical for full-size treebanks. As a tractable alternative, we consider an *implicit grammar* G^I (see Figure 1(b)) that has the same posterior probabilities as G. To construct G^I, we use a simplification of the PCFG-reduction of DOP by Goodman (1996a).[4] G^I has *base* symbols, which are the symbol types from the original treebank, as well as *indexed* symbols, which are obtained by assigning a unique index to each node token in the training treebank. The vast majority of symbols in G^I are therefore indexed symbols. While it may seem that such grammars will be overly large, they are in fact reasonably compact, being linear in the treebank size B, while G is exponential in the length of a sentence. In particular, we found that G^I was smaller than explicit extraction of all depth 1 and 2 unbinarized fragments for our

[4]The difference is that Goodman (1996a) collapses our BEGIN and END rules into the binary productions, giving a larger grammar which is less convenient for weighting.

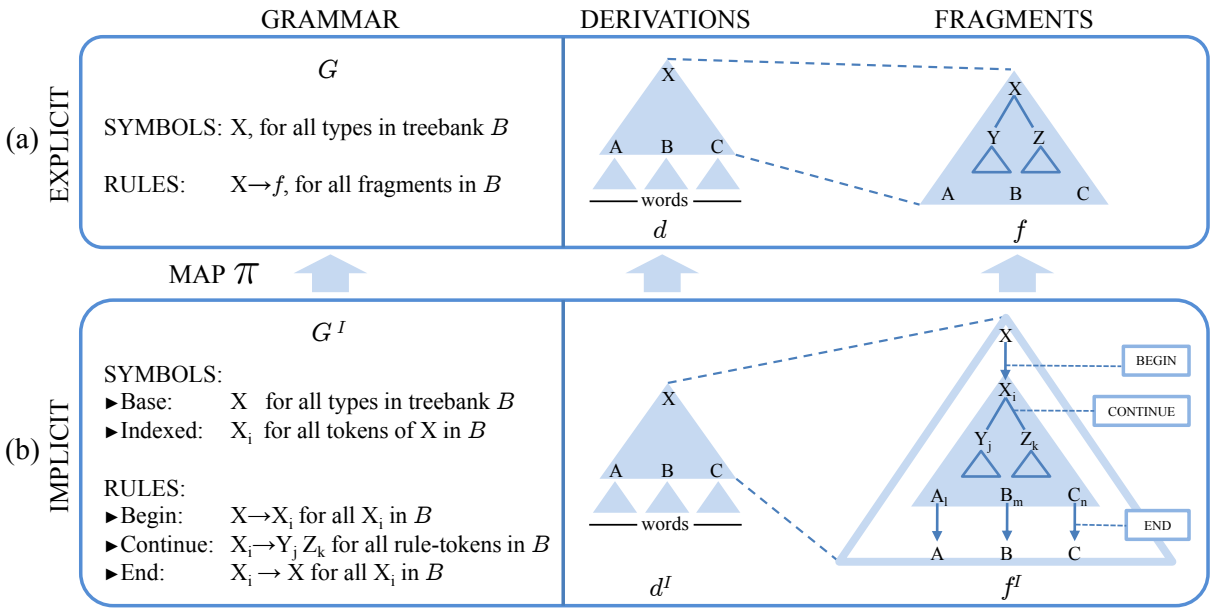


Figure 1: Grammar definition and sample derivations and fragments in the grammar for (a) the explicitly extracted all-fragments grammar G, and (b) its implicit representation G^I.

treebanks – in practice, even just the raw treebank grammar grows almost linearly in the size of B.[5]

There are 3 kinds of rules in G^I, which are illustrated in Figure 1(d). The BEGIN rules transition from a base symbol to an indexed symbol and represent the beginning of a fragment from G. The CONTINUE rules use only indexed symbols and correspond to specific depth-1 binary fragment tokens from training trees, representing the internal continuation of a fragment in G. Finally, END rules transition from an indexed symbol to a base symbol, representing the frontier of a fragment.

By construction, all derivations in G^I will segment, as shown in Figure 1(d), into regions corresponding to *tokens* of fragments from the training treebank B. Let π be the map which takes appropriate fragments in G^I (those that begin and end with base symbols and otherwise contain only indexed symbols), and maps them to the corresponding f in G. We can consider any derivation d^I in G^I to be a tree of fragments f^I, each fragment a token of a fragment type $f = \pi(f^I)$ in the original grammar G. By extension, we can therefore map any derivation d^I in G^I to the corresponding derivation $d = \pi(d^I)$ in G.

The mapping π is an onto mapping from G^I to G. In particular, each derivation d in G has a non-empty set of corresponding derivations $\{d^I\} = \pi^{-1}(d)$ in G^I, because fragments f in d correspond to multiple fragments f^I in G^I that differ only in their indexed symbols (one f^I per occurrence of f in B). Therefore, the set of derivations in G is preserved in G^I. We now discuss how weights can be preserved under π.

2.3 Equivalence for Weighted Grammars

In general, arbitrary weight functions ω on fragments in G do not decompose along the increased locality of G^I. However, we now consider a usefully broad class of weighting schemes for which the posterior probabilities under G of derivations d are preserved in G^I. In particular, assume that we have a weighting ω on rules in G^I which does not depend on the specific indices used. Therefore, any fragment f^I will have a weight in G^I of the form:

$$\omega_I(f^I) = \omega_{\text{BEGIN}}(b) \prod_{r \in \text{C}} \omega_{\text{CONT}}(r) \prod_{e \in \text{E}} \omega_{\text{END}}(e)$$

where b is the BEGIN rule, r are CONTINUE rules, and e are END rules in the fragment f^I (see Figure 1(d)). Because ω is assumed to not depend on the specific indices, all f^I which correspond to the same f under π will have the same weight $\omega_I(f)$ in G^I.

In this case, we can define an induced weight

[5] Just half the training set (19916 trees) itself had 1.7 million depth 1 and 2 unbinarized rules compared to the 0.9 million indexed symbols in G^I (after graph packing). Even extracting binarized fragments (depth 1 and 2, with one order of parent annotation) gives us 0.75 million rules, and, practically, we would need fragments of greater depth.

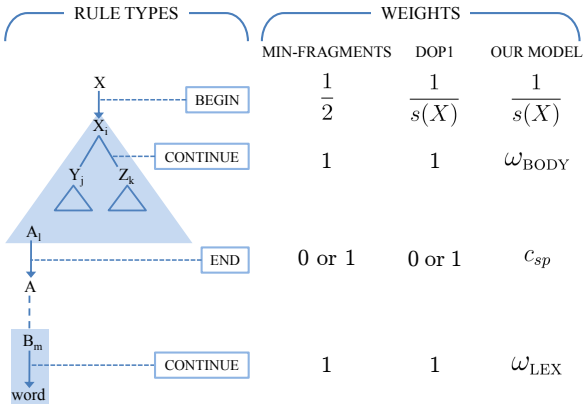


Figure 2: Rules defined for grammar G^I and weight schema for the DOP1 model, the Min-Fragments model (Goodman (2003)) and our model. Here $s(X)$ denotes the total number of fragments rooted at base symbol X.

for fragments f in G by

$$\omega_G(f) = \sum_{f^I \in \pi^{-1}(f)} \omega_I(f^I) = n(f)\omega_I(f)$$
$$= n(f)\omega_{\text{BEGIN}}(b') \prod_{r' \in \text{C}} \omega_{\text{CONT}}(r') \prod_{e' \in \text{E}} \omega_{\text{END}}(e')$$

where now b', r' and e' are non-indexed type abstractions of f's member productions in G^I and $n(f) = |\pi^{-1}(f)|$ is the number of tokens of f in B.

Under the weight function $\omega_G(f)$, any derivation d in G will have weight which obeys

$$\omega_G(d) = \prod_{f \in d} \omega_G(f) = \prod_{f \in d} n(f)\omega_I(f)$$
$$= \sum_{d^I \in d} \omega_I(d^I)$$

and so the posterior $P(d|s)$ of a derivation d for a sentence s will be the same whether computed in G or G^I. Therefore, provided our weighting function on fragments f in G decomposes over the derivational representation of f in G^I, we can equivalently compute the quantities we need for inference (see Section 4) using G^I instead.

3 Parameterization of Implicit Grammars

3.1 Classical DOP1

The original data-oriented parsing model 'DOP1' (Bod, 1993) is a particular instance of the general weighting scheme which decomposes appropriately over the implicit encoding, described in Section 2.3. Figure 2 shows rule weights for DOP1 in the parameter schema we have defined. The END rule weight is 0 or 1 depending on whether A is an intermediate symbol or not.[6] The local fragments in DOP1 were flat (non-binary) so this weight choice simulates that property by not allowing switching between fragments at intermediate symbols.

The original DOP1 model weights a fragment f in G as $\omega_G(f) = n(f)/s(X)$, i.e., the frequency of fragment f divided by the number of fragments rooted at base symbol X. This is simulated by our weight choices (Figure 2) where each fragment f^I in G^I has weight $\omega_I(f^I) = 1/s(X)$ and therefore, $\omega_G(f) = \sum_{f^I \in \pi^{-1}(f)} \omega_I(f^I) = n(f)/s(X)$. Given the weights used for DOP1, the recursive formula for the number of fragments $s(X_i)$ rooted at indexed symbol X_i (and for the CONTINUE rule $X_i \rightarrow Y_j\ Z_k$) is

$$s(X_i) = (1 + s(Y_j))(1 + s(Z_k)), \qquad (1)$$

where $s(Y_j)$ and $s(Z_k)$ are the number of fragments rooted at indexed symbols Y_j and Z_k (non-intermediate) respectively. The number of fragments $s(X)$ rooted at base symbol X is then $s(X) = \sum_{X_i} s(X_i)$.

Implicitly parsing with the full DOP1 model (no sampling of fragments) using the weights in Figure 2 gives a 68% parsing accuracy on the WSJ dev-set.[7] This result indicates that the weight of a fragment should depend on more than just its frequency.

3.2 Better Parameterization

As has been pointed out in the literature, large-fragment grammars can benefit from weights of fragments depending not only on their frequency but also on other properties. For example, Bod (2001) restricts the size and number of words in the frontier of the fragments, and Collins and Duffy (2002) and Goodman (2003) both give larger fragments smaller weights. Our model can incorporate both size and lexical properties. In particular, we set $\omega_{\text{CONT}}(r)$ for each binary CONTINUE rule r to a learned constant ω_{BODY}, and we set the weight for each rule with a POS parent to a

[6] Intermediate symbols are those created during binarization.

[7] For DOP1 experiments, we use no symbol refinement. We annotate with full left binarization history to imitate the flat nature of fragments in DOP1. We use mild coarse-pass pruning (Section 4.1) without which the basic all-fragments chart does not fit in memory. Standard WSJ treebank splits used: sec 2-21 training, 22 dev, 23 test.

Rule score: $r(A \rightarrow B\ C, i, k, j) = \sum_x \sum_y \sum_z O(A_x, i, j)\omega(A_x \rightarrow B_y\ C_z)I(B_y, i, k)I(C_z, k, j)$

Max-Constituent:	$q(A, i, j) = \frac{\sum_x O(A_x,i,j)I(A_x,i,j)}{\sum_r I(root_r,0,n)}$	$t_{max} = \operatorname{argmax}_t \sum_{c \in t} q(c)$
Max-Rule-Sum:	$q(A \rightarrow B\ C, i, k, j) = \frac{r(A \rightarrow B\ C,i,k,j)}{\sum_r I(root_r,0,n)}$	$t_{max} = \operatorname{argmax}_t \sum_{e \in t} q(e)$
Max-Variational:	$q(A \rightarrow B\ C, i, k, j) = \frac{r(A \rightarrow B\ C,i,k,j)}{\sum_x O(A_x,i,j)I(A_x,i,j)}$	$t_{max} = \operatorname{argmax}_t \prod_{e \in t} q(e)$

Figure 3: Inference: Different objectives for parsing with posteriors. A, B, C are base symbols, A_x, B_y, C_z are indexed symbols and i,j,k are between-word indices. Hence, (A_x, i, j) represents a constituent labeled with A_x spanning words i to j. $I(A_x, i, j)$ and $O(A_x, i, j)$ denote the inside and outside scores of this constituent, respectively. For brevity, we write $c \equiv (A, i, j)$ and $e \equiv (A \rightarrow B\ C, i, k, j)$. Also, t_{max} is the highest scoring parse. Adapted from Petrov and Klein (2007).

constant ω_{LEX} (see Figure 2). Fractional values of these parameters allow the weight of a fragment to depend on its size and lexical properties.

Another parameter we introduce is a 'switching-penalty' c_{sp} for the END rules (Figure 2). The DOP1 model uses binary values (0 if symbol is intermediate, 1 otherwise) as the END rule weight, which is equivalent to prohibiting fragment switching at intermediate symbols. We learn a fractional constant a_{sp} that allows (but penalizes) switching between fragments at annotated symbols through the formulation $c_{sp}(X_{intermediate}) = 1 - a_{sp}$ and $c_{sp}(X_{non-intermediate}) = 1 + a_{sp}$. This feature allows fragments to be assigned weights based on the binarization status of their nodes.

With the above weights, the recursive formula for $s(X_i)$, the total weighted number of fragments rooted at indexed symbol X_i, is different from DOP1 (Equation 1). For rule $X_i \rightarrow Y_j\ Z_k$, it is

$$s(X_i) = \omega_{\text{BODY}}.(c_{sp}(Y_j)+s(Y_j))(c_{sp}(Z_k)+s(Z_k)).$$

The formula uses ω_{LEX} in place of ω_{BODY} if r is a lexical rule (Figure 2).

The resulting grammar is primarily parameterized by the training treebank B. However, each setting of the hyperparameters $(\omega_{\text{BODY}}, \omega_{\text{LEX}}, a_{sp})$ defines a different conditional distribution on trees. We choose amongst these distributions by directly optimizing parsing F1 on our development set. Because this objective is not easily differentiated, we simply perform a grid search on the three hyperparameters. The tuned values are $\omega_{\text{BODY}} = 0.35$, $\omega_{\text{LEX}} = 0.25$ and $a_{sp} = 0.018$. For generalization to a larger parameter space, we would of course need to switch to a learning approach that scales more gracefully in the number of tunable hyperparameters.[8]

[8]Note that there has been a long history of DOP estimators. The generative DOP1 model was shown to be inconsistent by Johnson (2002). Later, Zollmann and Sima'an (2005) presented a statistically consistent estimator, with the basic insight of optimizing on a held-out set. Our estimator is not intended to be viewed as a generative model of trees at all, but simply a loss-minimizing conditional distribution within our parametric family.

	dev ($\leq$ 40)		test ($\leq$ 40)		test (all)	
Model	F1	EX	F1	EX	F1	EX
Constituent	**88.4**	33.7	**88.5**	33.0	**87.6**	30.8
Rule-Sum	88.2	**34.6**	88.3	33.8	87.4	31.6
Variational	87.7	34.4	87.7	**33.9**	86.9	**31.6**

Table 1: All-fragments WSJ results (accuracy F1 and exact match EX) for the constituent, rule-sum and variational objectives, using parent annotation and one level of markovization.

4 Efficient Inference

The previously described implicit grammar G^I defines a posterior distribution $P(d^I|s)$ over a sentence s via a large, indexed PCFG. This distribution has the property that, when marginalized, it is equivalent to a posterior distribution $P(d|s)$ over derivations in the correspondingly-weighted all-fragments grammar G. However, even with an explicit representation of G, we would not be able to tractably compute the parse that maximizes $P(t|s) = \sum_{d \in t} P(d|s) = \sum_{d^I \in t} P(d^I|s)$ (Sima'an, 1996). We therefore approximately maximize over trees by computing various existing approximations to $P(t|s)$ (Figure 3). Goodman (1996b), Petrov and Klein (2007), and Matsuzaki et al. (2005) describe the details of constituent, rule-sum and variational objectives respectively. Note that all inference methods depend on the posterior $P(t|s)$ only through marginal expectations of labeled constituent counts and anchored local binary tree counts, which are easily computed from $P(d^I|s)$ and equivalent to those from $P(d|s)$. Therefore, no additional approximations are made in G^I over G.

As shown in Table 1, our model (an all-fragments grammar with the weighting scheme

shown in Figure 2) achieves an accuracy of 88.5% (using simple parent annotation) which is 4-5% (absolute) better than the recent TSG work (Zuidema, 2007; Cohn et al., 2009; Post and Gildea, 2009) and also approaches state-of-the-art refinement-based parsers (e.g., Charniak and Johnson (2005), Petrov and Klein (2007)).[9]

4.1 Coarse-to-Fine Inference

Coarse-to-fine inference is a well-established way to accelerate parsing. Charniak et al. (2006) introduced multi-level coarse-to-fine parsing, which extends the basic pre-parsing idea by adding more rounds of pruning. Their pruning grammars were coarse versions of the raw treebank grammar. Petrov and Klein (2007) propose a multi-stage coarse-to-fine method in which they construct a sequence of increasingly refined grammars, reparsing with each refinement. In particular, in their approach, which we adopt here, coarse-to-fine pruning is used to quickly compute approximate marginals, which are then used to prune subsequent search. The key challenge in coarse-to-fine inference is the construction of coarse models which are much smaller than the target model, yet whose posterior marginals are close enough to prune with safely.

Our grammar G^I has a very large number of indexed symbols, so we use a coarse pass to prune away their unindexed abstractions. The simple, intuitive, and effective choice for such a coarse grammar G^C is a minimal PCFG grammar composed of the base treebank symbols X and the minimal depth-1 binary rules $X \rightarrow Y\ Z$ (and with the same level of annotation as in the full grammar). If a particular base symbol X is pruned by the coarse pass for a particular span (i, j) (i.e., the posterior marginal $P(X, i, j|s)$ is less than a certain threshold), then in the full grammar G^I, we do not allow building any indexed symbol X_l of type X for that span. Hence, the projection map for the coarse-to-fine model is π^C : $X_l\ (indexed\ symbol) \rightarrow X\ (base\ symbol)$.

We achieve a substantial improvement in speed and memory-usage from the coarse-pass pruning. Speed increases by a factor of 40 and memory-usage decreases by a factor of 10 when we go from no pruning to pruning with a -6.2 log posterior threshold.[10] Figure 4 depicts the variation in parsing accuracies in response to the amount of pruning done by the coarse-pass. Higher posterior pruning thresholds induce more aggressive pruning. Here, we observe an effect seen in previous work (Charniak et al. (1998), Petrov and Klein (2007), Petrov et al. (2008)), that a certain amount of pruning helps accuracy, perhaps by promoting agreement between the coarse and full grammars (model intersection). However, these 'fortuitous' search errors give only a small improvement and the peak accuracy is almost equal to the parsing accuracy without any pruning (as seen in Figure 5).[11] This outcome suggests that the coarse-pass pruning is critical for tractability but not for performance.

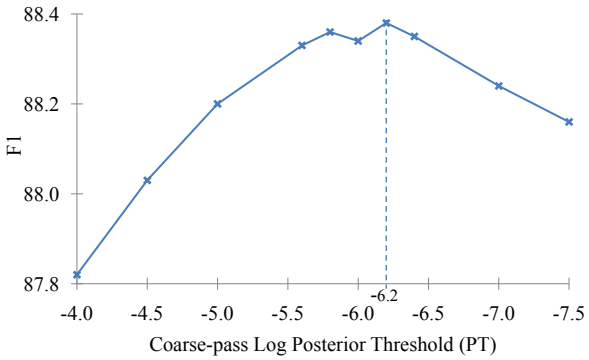


Figure 4: Effect of coarse-pass pruning on parsing accuracy (for WSJ dev-set, $\leq$ 40 words). Pruning increases to the left as log posterior threshold (PT) increases.

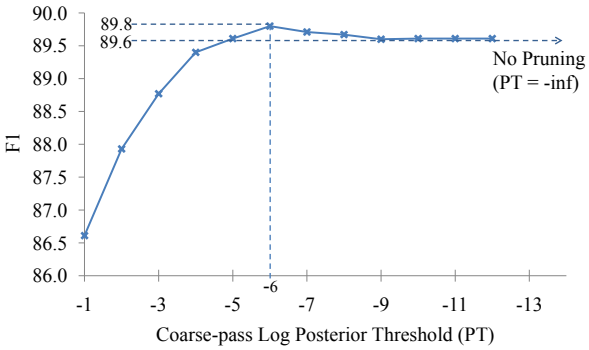


Figure 5: Effect of coarse-pass pruning on parsing accuracy (WSJ, training $\leq$ 20 words, tested on dev-set $\leq$ 20 words). This graph shows that the fortuitous improvement due to pruning is very small and that the peak accuracy is almost equal to the accuracy without pruning (the dotted line).

[9] All our experiments use the constituent objective except when we report results for max-rule-sum and max-variational parsing (where we use the parameters tuned for max-constituent, therefore they unsurprisingly do not perform as well as max-constituent). Evaluations use EVALB, see `http://nlp.cs.nyu.edu/evalb/`.

[10] Unpruned experiments could not be run for 40-word test sentences even with 50GB of memory, therefore we calculated the improvement factors using a smaller experiment with full training and sixty 30-word test sentences.

[11] To run experiments without pruning, we used training and dev sentences of length $\leq$ 20 for the graph in Figure 5.

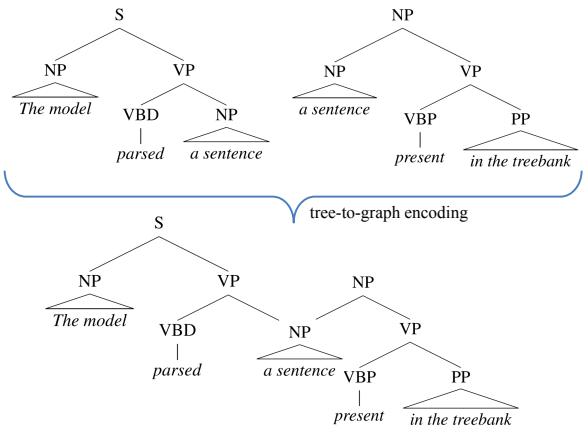


Figure 6: Collapsing the duplicate training subtrees converts them to a graph and reduces the number of indexed symbols significantly.

Parsing Model	No. of Indexed Symbols
Word-level Trees	1,900,056
Word-level Graph	903,056
Character-level Trees	12,280,848
Character-level Graph	1,109,399

Table 2: Number of indexed symbols for word-level and character-level parsing and their graph versions (for all-fragments grammar with parent annotation and one level of markovization).

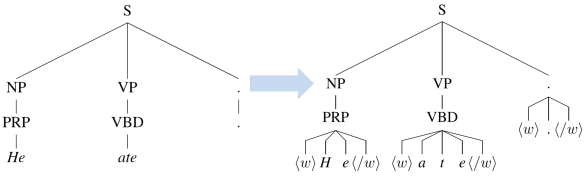


Figure 7: Character-level parsing: treating the sentence as a string of characters instead of words.

4.2 Packed Graph Encoding

The implicit all-fragments approach (Section 2.2) avoids explicit extraction of all rule fragments. However, the number of indexed symbols in our implicit grammar G^I is still large, because every node in each training tree (i.e., every symbol token) has a unique indexed symbol. We have around 1.9 million indexed symbol tokens in the word-level parsing model (this number increases further to almost 12.3 million when we parse character strings in Section 5.1). This large symbol space makes parsing slow and memory-intensive.

We reduce the number of symbols in our implicit grammar G^I by applying a compact, packed graph encoding to the treebank training trees. We collapse the duplicate *subtrees* (fragments that bottom out in terminals) over all training trees. This keeps the grammar unchanged because in an tree-substitution grammar, a node is defined (identified) by the subtree below it. We maintain a hashmap on the subtrees which allows us to easily discover the duplicates and bin them together. The collapsing converts all the training trees in the treebank to a graph with multiple parents for some nodes as shown in Figure 6. This technique reduces the number of indexed symbols significantly as shown in Table 2 (1.9 million goes down to 0.9 million, reduction by a factor of 2.1). This reduction increases parsing speed by a factor of 1.4 (and by a factor of 20 for character-level parsing, see Section 5.1) and reduces memory usage to under 4GB.

We store the duplicate-subtree counts for each indexed symbol of the collapsed graph (using a hashmap). When calculating the number of fragments $s(X_i)$ parented by an indexed symbol X_i (see Section 3.2), and when calculating the inside and outside scores during inference, we account for the collapsed subtree tokens by expanding the counts and scores using the corresponding multiplicities. Therefore, we achieve the compaction with negligible overhead in computation.

5 Improved Treebank Representations

5.1 Character-Level Parsing

The all-fragments approach to parsing has the added advantage that parsing below the word level requires no special treatment, i.e., we do not need an explicit lexicon when sentences are considered as strings of characters rather than words.

Unknown words in test sentences (unseen in training) are a major issue in parsing systems for which we need to train a complex lexicon, with various unknown classes or suffix tries. Smoothing factors need to be accounted for and tuned. With our implicit approach, we can avoid training a lexicon by building up the parse tree from characters instead of words. As depicted in Figure 7, each word in the training trees is split into its corresponding characters with start and stop boundary tags (and then binarized in a standard right-branching style). A test sentence's words are split up similarly and the test-parse is built from training fragments using the same model and inference procedure as defined for word-level parsing (see Sections 2, 3 and 4). The lexical items (alphabets, digits etc.) are now all known, so unlike word-level parsing, no sophisticated lexicon is needed.

We choose a slightly richer weighting scheme

Model	dev ($\leq$ 40) F1	dev ($\leq$ 40) EX	test ($\leq$ 40) F1	test ($\leq$ 40) EX	test (all) F1	test (all) EX
Constituent	**88.2**	33.6	**88.0**	31.9	**87.1**	29.8
Rule-Sum	88.0	33.9	87.8	**33.1**	87.0	**30.9**
Variational	87.6	**34.4**	87.2	32.3	86.4	30.2

Table 3: All-fragments WSJ results for the character-level parsing model, using parent annotation and one level of markovization.

Parsing Model	F1
No Refinement (P=0, H=0)*	71.3
Basic Refinement (P=1, H=1)*	80.0
All-Fragments + No Refinement (P=0, H=0)	85.7
All-Fragments + Basic Refinement (P=1, H=1)	88.4

Table 4: F1 for a basic PCFG, and incorporation of basic refinement, all-fragments and both, for WSJ dev-set ($\leq$ 40 words). $P = 1$ means parent annotation of all non-terminals, including the preterminal tags. $H = 1$ means one level of markovization. *Results from Klein and Manning (2003).

for this representation by extending the two-weight schema for CONTINUE rules (ω_{LEX} and ω_{BODY}) to a three-weight one: ω_{LEX}, ω_{WORD}, and ω_{SENT} for CONTINUE rules in the lexical layer, in the portion of the parse that builds words from characters, and in the portion of the parse that builds the sentence from words, respectively. The tuned values are $\omega_{\text{SENT}} = 0.35$, $\omega_{\text{WORD}} = 0.15$, $\omega_{\text{LEX}} = 0.95$ and $a_{sp} = 0$. The character-level model achieves a parsing accuracy of 88.0% (see Table 3), *despite lacking an explicit lexicon*.[12]

Character-level parsing expands the training trees (see Figure 7) and the already large indexed symbol space size explodes (1.9 million increases to 12.3 million, see Table 2). Fortunately, this is where the packed graph encoding (Section 4.2) is most effective because duplication of character strings is high (e.g., suffixes). The packing shrinks the symbol space size from 12.3 million to 1.1 million, a reduction by a factor of 11. This reduction increases parsing speed by almost a factor of 20 and brings down memory-usage to under 8GB.[13]

5.2 Basic Refinement: Parent Annotation and Horizontal Markovization

In a pure all-fragments approach, compositions of units which would have been independent in a basic PCFG are given joint scores, allowing the representation of certain non-local phenomena, such as lexical selection or agreement, which in fully local models require rich state-splitting or lexicalization. However, at substitution sites, the coarseness of raw unrefined treebank symbols still creates unrealistic factorization assumptions. A standard solution is symbol refinement; Johnson (1998) presents the particularly simple case of parent annotation, in which each node is marked with its parent in the underlying treebank. It is reasonable to hope that the gains from using large fragments and the gains from symbol refinement will be complementary. Indeed, previous work has shown or suggested this complementarity. Sima'an (2000) showed modest gains from enriching structural relations with semi-lexical (prehead) information. Charniak and Johnson (2005) showed accuracy improvements from composed local tree features on top of a lexicalized base parser. Zuidema (2007) showed a slight improvement in parsing accuracy when enough fragments were added to learn enrichments beyond manual refinements. Our work reinforces this intuition by demonstrating how complementary they are in our model ($\sim$20% error reduction on adding refinement to an all-fragments grammar, as shown in the last two rows of Table 4).

Table 4 shows results for a basic PCFG, and its augmentation with either basic refinement (parent annotation and one level of markovization), with all-fragments rules (as in previous sections), or both. The basic incorporation of large fragments alone does not yield particularly strong performance, nor does basic symbol refinement. However, the two approaches are quite additive in our model and combine to give nearly state-of-the-art parsing accuracies.

5.3 Additional Deterministic Refinement

Basic symbol refinement (parent annotation), in combination with all-fragments, gives test-set accuracies of 88.5% ($\leq$ 40 words) and 87.6% (all), shown as the Basic Refinement model in Table 5. Klein and Manning (2003) describe a broad set of simple, deterministic symbol refinements beyond parent annotation. We included ten of their simplest annotation features, namely: UNARY-DT, UNARY-RB, SPLIT-IN, SPLIT-AUX, SPLIT-CC, SPLIT-%, GAPPED-S, POSS-NP, BASE-NP and DOMINATES-V. None of these annotation schemes use any head information. This additional annotation (see Ad-

[12]Note that the word-level model yields a higher accuracy of 88.5%, but uses 50 complex unknown word categories based on lexical, morphological and position features (Petrov et al., 2006). Cohn et al. (2009) also uses this lexicon.

[13]Full char-level experiments (w/o packed graph encoding) could not be run even with 50GB of memory. We calculate the improvement factors using a smaller experiment with 70% training and fifty 20-word test sentences.

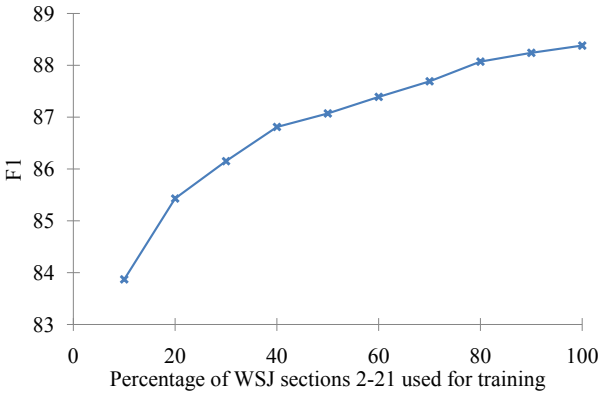


Figure 8: Parsing accuracy F1 on the WSJ dev-set ($\leq$ 40 words) increases with increasing percentage of training data.

Parsing Model	test ($\leq$ 40) F1	test ($\leq$ 40) EX	test (all) F1	test (all) EX
FRAGMENT-BASED PARSERS				
Zuidema (2007)	–	–	83.8*	26.9*
Cohn et al. (2009)	–	–	84.0	–
Post and Gildea (2009)	82.6	–	–	–
THIS PAPER				
All-Fragments				
+ Basic Refinement	88.5	33.0	87.6	30.8
+ Additional Refinement	88.7	33.8	88.1	31.7
REFINEMENT-BASED PARSERS				
Collins (1999)	88.6	–	88.2	–
Petrov and Klein (2007)	90.6	39.1	90.1	37.1

Table 5: Our WSJ test set parsing accuracies, compared to recent fragment-based parsers and top refinement-based parsers. Basic Refinement is our all-fragments grammar with parent annotation. Additional Refinement adds deterministic refinement of Klein and Manning (2003) (Section 5.3). *Results on the dev-set ($\leq$ 100).

ditional Refinement, Table 5) improves the test-set accuracies to 88.7% ($\leq$ 40 words) and 88.1% (all), which is equal to a strong lexicalized parser (Collins, 1999), even though our model does not use lexicalization or latent symbol-split induction.

6 Other Results

6.1 Parsing Speed and Memory Usage

The word-level parsing model using the whole training set (39832 trees, all-fragments) takes approximately 3 hours on the WSJ test set (2245 trees of $\leq$40 words), which is equivalent to roughly 5 seconds of parsing time per sentence; and runs in under 4GB of memory. The character-level version takes about twice the time and memory. This novel tractability of an all-fragments grammar is achieved using both coarse-pass pruning and packed graph encoding. Micro-optimization may further improve speed and memory usage.

6.2 Training Size Variation

Figure 8 shows how WSJ parsing accuracy increases with increasing amount of training data (i.e., percentage of WSJ sections 2-21). Even if we train on only 10% of the WSJ training data (3983 sentences), we still achieve a reasonable parsing accuracy of nearly 84% (on the development set, $\leq$ 40 words), which is comparable to the full-system results obtained by Zuidema (2007), Cohn et al. (2009) and Post and Gildea (2009).

6.3 Other Language Treebanks

On the French and German treebanks (using the standard dataset splits mentioned in Petrov and Klein (2008)), our simple all-fragments parser achieves accuracies in the range of top refinement-based parsers, even though the model parameters were tuned out of domain on WSJ. For German, our parser achieves an F1 of 79.8% compared to 81.5% by the state-of-the-art and substantially more complex Petrov and Klein (2008) work. For French, our approach yields an F1 of 78.0% vs. 80.1% by Petrov and Klein (2008).[14]

7 Conclusion

Our approach of using all fragments, in combination with basic symbol refinement, and even without an explicit lexicon, achieves results in the range of state-of-the-art parsers on full scale treebanks, across multiple languages. The main takeaway is that we can achieve such results in a very knowledge-light way with (1) no latent-variable training, (2) no sampling, (3) no smoothing beyond the existence of small fragments, and (4) no explicit unknown word model at all. While these methods offer a simple new way to construct an accurate parser, we believe that this general approach can also extend to other large-fragment tasks, such as machine translation.

Acknowledgments

This project is funded in part by BBN under DARPA contract HR0011-06-C-0022 and the NSF under grant 0643742.

[14] All results on the test set ($\leq$ 40 words).

References

Rens Bod. 1993. Using an Annotated Corpus as a Stochastic Grammar. In *Proceedings of EACL*.

Rens Bod. 2001. What is the Minimal Set of Fragments that Achieves Maximum Parse Accuracy? In *Proceedings of ACL*.

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

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-Based Best-First Chart Parsing. In *Proceedings of the 6th Workshop on Very Large Corpora*.

Eugene Charniak, Mark Johnson, et al. 2006. Multilevel Coarse-to-fine PCFG Parsing. In *Proceedings of HLT-NAACL*.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL*.

David Chiang. 2003. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Data-Oriented Parsing*.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of ACL*.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing Compact but Accurate Tree-Substitution Grammars. In *Proceedings of NAACL*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL*.

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. *Ph.D. thesis, University of Pennsylvania, Philadelphia*.

Steve Deneefe and Kevin Knight. 2009. Synchronous Tree Adjoining Machine Translation. In *Proceedings of EMNLP*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*.

Joshua Goodman. 1996a. Efficient Algorithms for Parsing the DOP Model. In *Proceedings of EMNLP*.

Joshua Goodman. 1996b. Parsing Algorithms and Metrics. In *Proceedings of ACL*.

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In *Bod R, Scha R, Sima'an K (eds.) Data-Oriented Parsing. University of Chicago Press, Chicago, IL*.

James Henderson. 2004. Discriminative Training of a Neural Network Statistical Parser. In *Proceedings of ACL*.

Mark Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24:613–632.

Mark Johnson. 2002. The DOP Estimation Method Is Biased and Inconsistent. In *Computational Linguistics 28(1)*.

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*.

Philipp Koehn, Franz Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*.

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

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL-HLT*.

Slav Petrov and Dan Klein. 2008. Sparse Multi-Scale Grammars for Discriminative Latent Variable Parsing. In *Proceedings of EMNLP*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING-ACL*.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-Fine Syntactic Machine Translation using Language Projections. In *Proceedings of EMNLP*.

Matt Post and Daniel Gildea. 2009. Bayesian Learning of a Tree Substitution Grammar. In *Proceedings of ACL-IJCNLP*.

Philip Resnik. 1992. Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing. In *Proceedings of COLING*.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In *R. de Kort and G.L.J. Leerdam (eds.): Computertoepassingen in de Neerlandistiek*.

Khalil Sima'an. 1996. Computational Complexity of Probabilistic Disambiguation by means of Tree-Grammars. In *Proceedings of COLING*.

Khalil Sima'an. 2000. Tree-gram Parsing: Lexical Dependencies and Structural Relations. In *Proceedings of ACL*.

Andreas Zollmann and Khalil Sima'an. 2005. A Consistent and Efficient Estimator for Data-Oriented Parsing. *Journal of Automata, Languages and Combinatorics (JALC)*, 10(2/3):367–388.

Willem Zuidema. 2007. Parsimonious Data-Oriented Parsing. In *Proceedings of EMNLP-CoNLL*.

Joint Syntactic and Semantic Parsing of Chinese

Junhui Li and **Guodong Zhou**
School of Computer Science & Technology
Soochow University
Suzhou, China 215006
{lijunhui, gdzhou}@suda.edu.cn

Hwee Tou Ng
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417
nght@comp.nus.edu.sg

Abstract

This paper explores joint syntactic and semantic parsing of Chinese to further improve the performance of both syntactic and semantic parsing, in particular the performance of semantic parsing (in this paper, semantic role labeling). This is done from two levels. Firstly, an integrated parsing approach is proposed to integrate semantic parsing into the syntactic parsing process. Secondly, semantic information generated by semantic parsing is incorporated into the syntactic parsing model to better capture semantic information in syntactic parsing. Evaluation on Chinese TreeBank, Chinese PropBank, and Chinese NomBank shows that our integrated parsing approach outperforms the pipeline parsing approach on n-best parse trees, a natural extension of the widely used pipeline parsing approach on the top-best parse tree. Moreover, it shows that incorporating semantic role-related information into the syntactic parsing model significantly improves the performance of both syntactic parsing and semantic parsing. To our best knowledge, this is the first research on exploring syntactic parsing and semantic role labeling for both verbal and nominal predicates in an integrated way.

1 Introduction

Semantic parsing maps a natural language sentence into a formal representation of its meaning. Due to the difficulty in deep semantic parsing, most previous work focuses on shallow semantic parsing, which assigns a simple structure (such as WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW) to each predicate in a sentence. In particular, the well-defined semantic role labeling (SRL) task has been drawing increasing attention in recent years due to its importance in natural language processing (NLP) applications, such as question answering (Narayanan and Harabagiu, 2004), information extraction (Surdeanu et al., 2003), and co-reference resolution (Kong et al., 2009). Given a sentence and a predicate (either a verb or a noun) in the sentence, SRL recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate. In both English and Chinese PropBank (Palmer et al., 2005; Xue and Palmer, 2003), and English and Chinese NomBank (Meyers et al., 2004; Xue, 2006), these semantic arguments include core arguments (e.g., Arg0 for agent and Arg1 for recipient) and adjunct arguments (e.g., ArgM-LOC for locative argument and ArgM-TMP for temporal argument). According to predicate type, SRL can be divided into SRL for verbal predicates (verbal SRL, in short) and SRL for nominal predicates (nominal SRL, in short).

With the availability of large annotated corpora such as FrameNet (Baker et al., 1998), PropBank, and NomBank in English, data-driven techniques, including both feature-based and kernel-based methods, have been extensively studied for SRL (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005; Pradhan et al., 2005; Liu and Ng, 2007). Nevertheless, for both verbal and nominal SRL, state-of-the-art systems depend heavily on the top-best parse tree and there exists a large performance gap between SRL based on the gold parse tree and the top-best parse tree. For example, Pradhan et al. (2005) suffered a performance drop of 7.3 in F1-measure on English PropBank when using the top-best parse tree returned from Charniak's parser (Charniak, 2001). Liu and Ng (2007) reported a performance drop of 4.21 in F1-measure on English NomBank.

Compared with English SRL, Chinese SRL suffers more seriously from syntactic parsing. Xue (2008) evaluated on Chinese PropBank and showed that the performance of Chinese verbal SRL drops by about 25 in F1-measure when replacing gold parse trees with automatic ones. Likewise, Xue (2008) and Li et al. (2009) reported a performance drop of about 12 in F1-measure in Chinese NomBank SRL.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1108–1117,
Uppsala, Sweden, 11-16 July 2010.

While it may be difficult to further improve syntactic parsing, a promising alternative is to perform both syntactic and semantic parsing in an integrated way. Given the close interaction between the two tasks, joint learning not only allows uncertainty about syntactic parsing to be carried forward to semantic parsing but also allows useful information from semantic parsing to be carried backward to syntactic parsing.

This paper explores joint learning of syntactic and semantic parsing for Chinese texts from two levels. Firstly, an integrated parsing approach is proposed to benefit from the close interaction between syntactic and semantic parsing. This is done by integrating semantic parsing into the syntactic parsing process. Secondly, various semantic role-related features are directly incorporated into the syntactic parsing model to better capture semantic role-related information in syntactic parsing. Evaluation on Chinese TreeBank, Chinese PropBank, and Chinese NomBank shows that our method significantly improves the performance of both syntactic and semantic parsing. This is promising and encouraging. To our best knowledge, this is the first research on exploring syntactic parsing and SRL for verbal and nominal predicates in an integrated way.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents our baseline systems for syntactic and semantic parsing. Section 4 presents our proposed method of joint syntactic and semantic parsing for Chinese texts. Section 5 presents the experimental results. Finally, Section 6 concludes the paper.

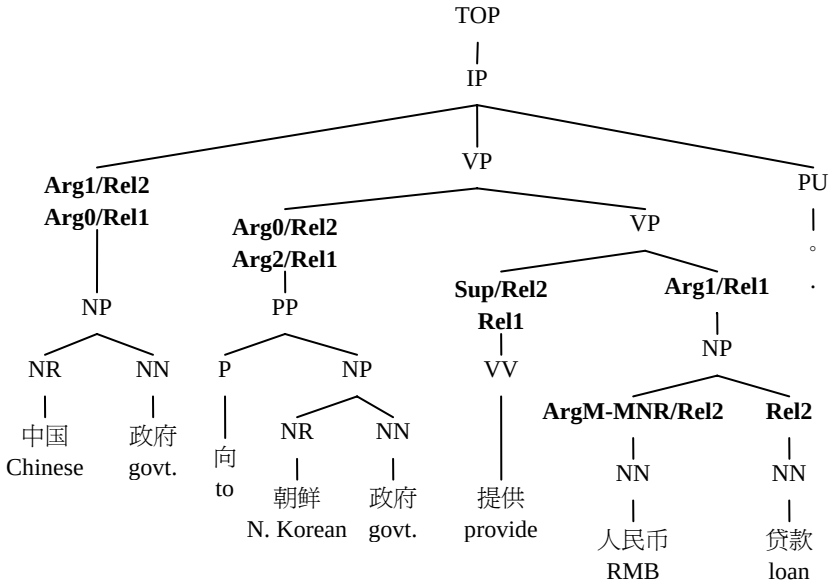
2 Related Work

Compared to the large body of work on either syntactic parsing (Ratnaparkhi, 1999; Collins, 1999; Charniak, 2001; Petrov and Klein, 2007), or SRL (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005; Jiang and Ng, 2006), there is relatively less work on their joint learning.

Koomen et al. (2005) adopted the outputs of multiple SRL systems (each on a single parse tree) and combined them into a coherent predicate argument output by solving an optimization problem. Sutton and McCallum (2005) adopted a probabilistic SRL system to re-rank the N-best results of a probabilistic syntactic parser. However, they reported negative results, which they blamed on the inaccurate probability estimates from their locally trained SRL model.

As an alternative to the above pseudo-joint learning methods (strictly speaking, they are still pipeline methods), one can augment the syntactic label of a constituent with semantic information, like what function parsing does (Merlo and Musillo, 2005). Yi and Palmer (2005) observed that the distributions of semantic labels could potentially interact with the distributions of syntactic labels and redefined the boundaries of constituents. Based on this observation, they incorporated semantic role information into syntactic parse trees by extending syntactic constituent labels with their coarse-grained semantic roles (core argument or adjunct argument) in the sentence, and thus unified semantic parsing and syntactic parsing. The actual fine-grained semantic roles are assigned, as in other methods, by an ensemble classifier. However, the results obtained with this method were negative, and they concluded that semantic parsing on PropBank was too difficult due to the differences between chunk annotation and tree structure. Motivated by Yi and Palmer (2005), Merlo and Musillo (2008) first extended a statistical parser to produce a richly annotated tree that identifies and labels nodes with semantic role labels as well as syntactic labels. Then, they explored both rule-based and machine learning techniques to extract predicate-argument structures from this enriched output. Their experiments showed that their method was biased against these roles in general, thus lowering recall for them (e.g., precision of 87.6 and recall of 65.8).

There have been other efforts in NLP on joint learning with various degrees of success. In particular, the recent shared tasks of CoNLL 2008 and 2009 (Surdeanu et al., 2008; Hajic et al., 2009) tackled joint parsing of syntactic and semantic dependencies. However, all the top 5 reported systems decoupled the tasks, rather than building joint models. Compared with the disappointing results of joint learning on syntactic and semantic parsing, Miller et al. (2000) and Finkel and Manning (2009) showed the effectiveness of joint learning on syntactic parsing and some simple NLP tasks, such as information extraction and name entity recognition. In addition, attempts on joint Chinese word segmentation and part-of-speech (POS) tagging (Ng and Low, 2004; Zhang and Clark, 2008) also illustrate the benefits of joint learning.



Chinese government provides RMB loan to North Korean government.

Figure 1: Two predicates (Rel1 and Rel2) and their arguments in the style of Chinese PropBank and NomBank.

3 Baseline: Pipeline Parsing on Top-Best Parse Tree

In this section, we briefly describe our approach to syntactic parsing and semantic role labeling, as well as the baseline system with pipeline parsing on the top-best parse tree.

3.1 Syntactic Parsing

Our syntactic parser re-implements Ratnaparkhi (1999), which adopts the maximum entropy principle. The parser recasts a syntactic parse tree as a sequence of decisions similar to those of a standard shift-reduce parser and the parsing process is organized into three left-to-right passes via four procedures, called TAG, CHUNK, BUILD, and CHECK.

First pass. The first pass takes a tokenized sentence as input, and uses TAG to assign each word a part-of-speech.

Second pass. The second pass takes the output of the first pass as input, and uses CHUNK to recognize basic chunks in the sentence.

Third pass. The third pass takes the output of the second pass as input, and always alternates between BUILD and CHECK in structural parsing in a recursive manner. Here, BUILD decides whether a subtree will start a new constituent or join the incomplete constituent immediately to its left. CHECK finds the most recently proposed constituent, and decides if it is complete.

3.2 Semantic Role Labeling

Figure 1 demonstrates an annotation example of Chinese PropBank and NomBank. In the figure, the verbal predicate "提供/provide" is annotated with three core arguments (i.e., "NP (中国/Chinese 政府/govt.)" as Arg0, "PP (向/to 朝鲜/N. Korean 政府/govt.)" as Arg2, and "NP (人民币/RMB 贷款/loan)" as Arg1), while the nominal predicate "贷款/loan" is annotated with two core arguments (i.e., "NP (中国/Chinese 政府/govt.)" as Arg1 and "PP (向/to 朝鲜/N. Korean 政府/govt.)" as Arg0), and an adjunct argument (i.e., "NN (人民币/RMB)" as ArgM-MNR, denoting the manner of loan). It is worth pointing out that there is a (Chinese) NomBank-specific label in Figure 1, Sup (support verb) (Xue, 2006), to help introduce the arguments which occur outside the nominal predicate-headed noun phrase. In (Chinese) NomBank, a verb is considered to be a support verb only if it shares at least an argument with the nominal predicate.

3.2.1 Automatic Predicate Recognition

Automatic predicate recognition is a prerequisite for the application of SRL systems. For verbal predicates, it is very easy. For example, 99% of verbs are annotated as predicates in Chinese PropBank. Therefore, we can simply select any word with a part-of-speech (POS) tag of VV, VA, VC, or VE as verbal predicate.

Unlike verbal predicate recognition, nominal predicate recognition is quite complicated. For

example, only 17.5% of nouns are annotated as predicates in Chinese NomBank. It is quite common that a noun is annotated as a predicate in some cases but not in others. Therefore, automatic predicate recognition is vital to nominal SRL. In principle, automatic predicate recognition can be cast as a binary classification (e.g., Predicate vs. Non-Predicate) problem. For nominal predicates, a binary classifier is trained to predict whether a noun is a nominal predicate or not. In particular, any word POS-tagged as NN is considered as a predicate candidate in both training and testing processes. Let the nominal predicate candidate be w_0, and its left and right neighboring words/POSs be $w_{-1}/p_{-1}$and w_1/p_1, respectively. Table 1 lists the feature set used in our model. In Table 1, local features present the candidate's contextual information while global features show its statistical information in the whole training set.

Type	Description
local features	w_0, w_{-1}, w_1, p_{-1}, p_1 The first and last characters of the candidate
global features	Whether w_0 is ever tagged as a verb in the training data? Yes/No Whether w_0 is ever annotated as a nominal predicate in the training data? Yes/No The most likely label for w_0 when it occurs together with w_{-1} and w_1. The most likely label for w_0 when it occurs together with w_{-1}. The most likely label for w_0 when it occurs together with w_1.

Table 1: Feature set for nominal predicate recognition

3.2.2 SRL for Chinese Predicates

Our Chinese SRL models for both verbal and nominal predicates adopt the widely-used SRL framework, which divides the task into three sequential sub-tasks: argument pruning, argument identification, and argument classification. In particular, we follow Xue (2008) and Li et al. (2009) to develop verbal and nominal SRL models, respectively. Moreover, we have further improved the performance of Chinese verbal SRL by exploring additional features, e.g., voice position that indicates the voice maker (BA, BEI) is before or after the constituent in focus, the rule that expands the parent of the constituent in focus, and the core arguments defined in the predicate's frame file. For nominal SRL, we simply use the final feature set of Li et al. (2009). As a result, our Chinese verbal and nominal SRL systems achieve performance of 92.38 and 72.67 in F1-measure respectively (on golden parse trees and golden predicates), which are comparable to Xue (2008) and Li et al. (2009). For more details, please refer to Xue (2008) and Li et al. (2009).

3.3 Pipeline Parsing on Top-best Parse Tree

Similar to most of the state-of-the-art systems (Pradhan et al., 2005; Xue, 2008; Li et al., 2009), the top-best parse tree is first returned from our syntactic parser and then fed into the SRL system. Specifically, the verbal (nominal) SRL labeler is in charge of verbal (nominal) predicates, respectively. For each sentence, since SRL is only performed on one parse tree, only constituents in it are candidates for semantic arguments. Therefore, if no constituent in the parse tree can map the same text span to an argument in the manual annotation, the system will not get a correct annotation.

4 Joint Syntactic and Semantic Parsing

In this section, we first explore pipeline parsing on N-best parse trees, as a natural extension of pipeline parsing on the top-best parse tree. Then, joint syntactic and semantic parsing is explored for Chinese texts from two levels. Firstly, an integrated parsing approach to joint syntactic and semantic parsing is proposed. Secondly, various semantic role-related features are directly incorporated into the syntactic parsing model for better interaction between the two tasks.

4.1 Pipeline Parsing on N-best Parse Trees

The pipeline parsing approach employed in this paper is largely motivated by the general framework of re-ranking, as proposed in Sutton and McCallum (2005). The idea behind this approach is that it allows uncertainty about syntactic parsing to be carried forward through an N-best list, and that a reliable SRL system, to a certain extent, can reflect qualities of syntactic parse trees. Given a sentence *x*, a joint parsing model is defined over a semantic frame *F* and a parse tree *t* in a log-linear way:

$$\begin{aligned} &Score(F,t \mid x) \\ &= (1-\alpha)\log P(F \mid t,x) + \alpha \log P(t \mid x) \end{aligned} \quad (1)$$

where *P(t|x)* is returned by a probabilistic syntactic parsing model, e.g., our syntactic parser, and *P(F|t, x)* is returned by a probabilistic semantic parsing model, e.g. our verbal & nominal

Algorithm 1. The algorithm integrating syntactic parsing and SRL.

```
Assume:
  t: constituent which is complete with "YES" decision of CHECK procedure
  P: number of predicates
  P_i: i^th predicate
  S: SRL result, set of predicates and its arguments
BEGIN
  srl_prob = 0.0;
  FOR i=1 to P DO
    IF t covers P_i THEN
      T = number of children of t;
      FOR j=1 to T DO
        IF t's j^th child Ch_j does not cover P_i THEN
          Run SRL given predicate P_i and constituent Ch_j to get their semantic role
            lbl and its probability prob;
          IF lbl does not indicate non-argument THEN
            srl_prob += log( prob );
            S = S ∪ {(P_i, Ch_j, lbl)};
          END IF
        END IF
      END FOR
    END IF
  END FOR
  return srl_prob;
END
```

SRL systems. In our pipeline parsing approach, $P(t|x)$ is calculated as the product of all involved decisions' probabilities in the syntactic parsing model, and $P(F|t, x)$ is calculated as the product of all the semantic role labels' probabilities in a sentence (including both verbal and nominal SRL). That is to say, we only consider those constituents that are supposed to be arguments. Here, the parameter α is a balance factor indicating the importance of the semantic parsing model.

In particular, (F^*, t^*) with maximal $Score(F, t|x)$ is selected as the final syntactic and semantic parsing results. Given a sentence, N-best parse trees are generated first using the syntactic parser, and then for each parse tree, we predict the best SRL frame using our verbal and nominal SRL systems.

4.2 Integrated Parsing

Although pipeline parsing on N-best parse trees could relieve severe dependence on the quality of the top-best parse tree, there is still a potential drawback: this method suffers from the limited scope covered by the N-best parse trees since the items in the parse tree list may be too similar, especially for long sentences. For example, 50-best parse trees can only represent a combination of 5 to 6 binary ambiguities since 2^5 < 50 < 2^6.

Ideally, we should perform SRL on as many parse trees as possible, so as to enlarge the search scope. However, pipeline parsing on all possible parse trees is time-consuming and thus unrealistic. As an alternative, we turn to integrated parsing, which aims to perform syntactic and semantic parsing synchronously. The key idea is to construct a parse tree in a bottom-up way so that it is feasible to perform SRL at suitable moments, instead of only when the whole parse tree is built. Integrated parsing is practicable, mostly due to the following two observations: (1) Given a predicate in a parse tree, its semantic arguments are usually siblings of the predicate, or siblings of its ancestor. Actually, this special observation has been widely employed in SRL to prune non-arguments for a verbal or nominal predicate (Xue, 2008; Li et al., 2009). (2) SRL feature spaces (both in feature-based method and kernel-based method) mostly focus on the predicate-argument structure of a given (predicate, argument) pair. That is to say, once a predicate-argument structure is formed (i.e., an argument candidate is connected with the given predicate), there is enough contextual information to predict their SRL relation.

As far as our syntactic parser is concerned, we invoke the SRL systems once a new constituent covering a predicate is complete with a "YES" decision from the CHECK procedure. Algorithm

1 illustrates the integration of syntactic and semantic parsing. For the example shown in Figure 2, the CHECK procedure predicts a "YES" decision, indicating the immediately proposed constituent "VP (提供/provide 人民币/RMB 贷款/loan)" is complete. So, at this moment, the verbal SRL system is invoked to predict the semantic label of the constituent "NP (人民币/RMB 贷款/loan)", given the verbal predicate "VV (提供/provide)". Similarly, "PP (向/to 朝鲜/N. Korean 政府/govt.)" would also be semantically labeled as soon as "PP (向/to 朝鲜/N. Korean 政府/govt.)" and "VP (提供/provide 人民币/RMB 贷款/loan)" are merged into a bigger VP. In this way, both syntactic and semantic parsing are accomplished when the root node TOP is formed. It is worth pointing out that all features (Xue, 2008; Li et al., 2009) used in our SRL model can be instantiated and their values are same as the ones when the whole tree is available. In particular, the probability computed from the SRL model is interpolated with that of the syntactic parsing model in a log-linear way (with equal weights in our experiments). This is due to our hypothesis that the probability returned from SRL model is helpful to joint syntactic and semantic parsing, considering the close interaction between the two tasks.

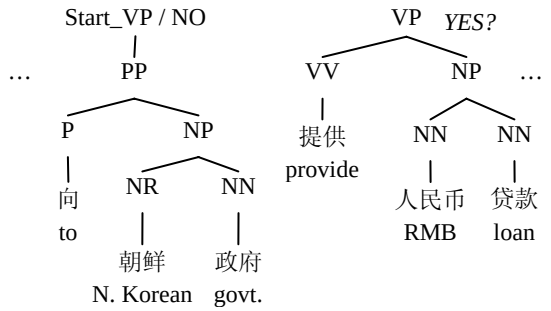


Figure 2: An application of CHECK with *YES* as the decision. Thus, VV (提供/provide) and NP (人民币/RMB 贷款/loan) reduce to a big VP.

4.3 Integrating Semantic Role-related Features into Syntactic Parsing Model

The integrated parsing approach as shown in Section 4.2 performs syntactic and semantic parsing synchronously. In contrast to traditional syntactic parsers where no semantic role-related information is used, it may be interesting to investigate the contribution of such information in the syntactic parsing model, due to the availability of such information in the syntactic parsing process. In addition, it is found that 11% of predicates in a sentence are speculatively attached with two or more core arguments with the same label due to semantic parsing errors (partly caused by syntactic parsing errors in automatic parse trees). This is abnormal since a predicate normally only allows at most one argument of each core argument role (i.e., Arg0-Arg4). Therefore, such syntactic errors should be avoidable by considering those arguments already obtained in the bottom-up parsing process. On the other hand, taking those expected semantic roles into account would help the syntactic parser. In terms of our syntactic parsing model, this is done by directly incorporating various semantic role-related features into the syntactic parsing model (i.e., the BUILD procedure) when the newly-formed constituent covers one or more predicates.

For the example shown in Figure 2, once the constituent "VP (提供/provide 人民币/RMB 贷款/loan)", which covers a verbal predicate "VV (提供/provide)", is complete, the verbal SRL model would be triggered first to mark constituent "NP (人民币/RMB 贷款/loan)" as ARG1, given predicate "VV (提供/provide)". Then, the BUILD procedure is called to make the BUILD decision for the newly-formed constituent "VP (提供/provide 人民币/RMB 贷款/loan)". Table 2 lists various semantic role-related features explored in our syntactic parsing model and their instantiations with regard to the example shown in Figure 2. In Table 2, feature sf4 gives the possible core semantic roles that the focus predicate may take, according to its frame file; feature sf5 presents the semantic roles that the focus predicate has already occupied; feature sf6 indicates the semantic roles that the focus predicate is expecting; and SF1-SF8 are combined features. Specifically, if the current constituent covers *n* predicates, then *14 * n* features would be instantiated. Moreover, we differentiate whether the focus predicate is verbal or nominal, and whether it is the head word of the current constituent.

Feature Selection. Some features proposed above may not be effective in syntactic parsing. Here we adopt the greedy feature selection algorithm as described in Jiang and Ng (2006) to select useful features empirically and incrementally according to their contributions on the development data. The algorithm repeatedly selects one feature each time which contributes the most, and stops when adding any of the remain-

ing features fails to improve the syntactic parsing performance.

Feat.	Description
sf1	Path: the syntactic path from C to P. *(VP>VV)*
sf2	Predicate: the predicate itself. *(提供/provide)*
sf3	Predicate class (Xue, 2008): the class that P belongs to. *(C3b)*
sf4	Possible roles: the core semantic roles P may take. *(Arg0, Arg1, Arg2)*
sf5	Detected roles: the core semantic roles already assigned to P. *(Arg1)*
sf6	Expected roles: possible semantic roles P is still expecting. *(Arg0, Arg2)*
SF1	For each already detected argument, its role label + its path from P. *(Arg1+VV<VP>NP)*
SF2	sf1 + sf2. *(VP>VV+提供/provide)*
SF3	sf1 + sf3. *(VP>VV+C3b)*
SF4	Combined possible argument roles. *(Arg0+Arg1+Arg2)*
SF5	Combined detected argument roles. *(Arg1)*
SF6	Combined expected argument roles. *(Arg0+Arg2)*
SF7	For each expected semantic role, sf1 + its role label. *(VP>VV+Arg0, VP>VV+Arg2)*
SF8	For each expected semantic role, sf2 + its role label. *(提供/provide+Arg0, 提供/provide+Arg2)*

Table 2: SRL-related features and their instantiations for syntactic parsing, with "VP (提供/provide 人民币/RMB 贷款/loan)" as the current constituent C and "提供/provide" as the focus predicate P, based on Figure 2.

5 Experiments and Results

We have evaluated our integrated parsing approach on Chinese TreeBank 5.1 and corresponding Chinese PropBank and NomBank.

5.1 Experimental Settings

This version of Chinese PropBank and Chinese NomBank consists of standoff annotations on the file (chtb 001 to 1151.fid) of Chinese Penn TreeBank 5.1. Following the experimental settings in Xue (2008) and Li et al. (2009), 648 files (chtb 081 to 899.fid) are selected as the training data, 72 files (chtb 001 to 040.fid and chtb 900 to 931.fid) are held out as the test data, and 40 files (chtb 041 to 080.fid) are selected as the development data. In particular, the training, test, and development data contain 31,361 (8,642), 3,599 (1,124), and 2,060 (731) verbal (nominal) propositions, respectively.

For the evaluation measurement on syntactic parsing, we report labeled recall, labeled precision, and their F1-measure. Also, we report recall, precision, and their F1-measure for evaluation of SRL on automatic predicates, combining verbal SRL and nominal SRL. An argument is correctly labeled if there is an argument in manual annotation with the same semantic label that spans the same words. Moreover, we also report the performance of predicate recognition. To see whether an improvement in F1-measure is statistically significant, we also conduct significance tests using a type of stratified shuffling which in turn is a type of compute-intensive randomized tests. In this paper, '>>>', '>>', and '>' denote p-values less than or equal to 0.01, in-between (0.01, 0.05], and bigger than 0.05, respectively.

We are not aware of any SRL system combing automatic predicate recognition, verbal SRL and nominal SRL on Chinese PropBank and NomBank. Xue (2008) experimented independently with verbal and nominal SRL and assumed correct predicates. Li et al. (2009) combined nominal predicate recognition and nominal SRL on Chinese NomBank. The CoNLL-2009 shared task (Hajic et al., 2009) included both verbal and nominal SRL on dependency parsing, instead of constituent-based syntactic parsing. Thus the SRL performances of their systems are not directly comparable to ours.

5.2 Results and Discussions

Results of pipeline parsing on N-best parse trees. While performing pipeline parsing on N-best parse trees, 20-best (the same as the heap size in our syntactic parsing) parse trees are obtained for each sentence using our syntactic parser as described in Section 3.1. The balance factor α is set to 0.5 indicating that the two components in formula (1) are equally important. Table 3 compares the two pipeline parsing approaches on the top-best parse tree and the N-best parse trees. It shows that the approach on N-best parse trees outperforms the one on the top-best parse tree by 0.42 (>>>) in F1-measure on SRL. In addition, syntactic parsing also benefits from the N-best parse trees approach with improvement of 0.17 (>>>) in F1-measure. This suggests that pipeline parsing on N-best parse trees can improve both syntactic and semantic parsing.

It is worth noting that our experimental results in applying the re-ranking framework in Chinese pipeline parsing on N-best parse trees are very encouraging, considering the pessimistic results of Sutton and McCallum (2005), in which the re-ranking framework failed to improve the performance on English SRL. It may be because,

unlike Sutton and McCallum (2005), $P(F, t|x)$ defined in this paper only considers those constituents which are identified as arguments. This can effectively avoid the noises caused by the predominant non-argument constituents. Moreover, the huge performance gap between Chinese semantic parsing on the gold parse tree and that on the top-best parse tree leaves much room for performance improvement.

Method	Task	R (%)	P (%)	F1
Pipeline on top -best parse tree	Syntactic	76.68	79.12	77.88
	SRL	62.96	65.04	63.98
	Predicate	94.18	92.28	93.22
	V-SRL	65.33	68.52	66.88
	V-Predicate	89.52	93.12	91.29
	N-SRL	49.58	48.19	48.88
	N-Predicate	86.83	71.76	78.58
Pipeline on 20 -best parse trees	Syntactic	76.89	79.25	78.05
	SRL	62.99	65.88	64.40
	Predicate	94.07	92.22	93.13
	V-SRL	65.41	69.09	67.20
	V-Predicate	89.66	93.02	91.31
	N-SRL	49.24	49.46	49.35
	N-Predicate	86.65	72.15	78.74
Integrated parsing	Syntactic	77.14	79.01	78.07
	SRL	62.67	67.67	65.07
	Predicate	93.97	92.42	93.19
	V-SRL	65.37	70.27	67.74
	V-Predicate	90.08	92.87	91.45
	N-SRL	48.02	52.83	50.31
	N-Predicate	85.41	73.23	78.85
Integrated parsing with semantic role-related features	Syntactic	77.47	79.58	78.51
	SRL	63.14	68.17	65.56
	Predicate	93.97	92.52	93.24
	V-SRL	65.74	70.98	68.26
	V-Predicate	89.86	93.17	91.49
	N-SRL	48.80	52.67	50.66
	N-Predicate	85.85	72.78	78.78

Table 3: Syntactic and semantic parsing performance on test data (using gold standard word boundaries). "V-" denotes "verbal" while "N-"denotes "nominal".

Results of integrated parsing. Table 3 also compares the integrated parsing approach with the two pipeline parsing approaches. It shows that the integrated parsing approach improves the performance of both syntactic and semantic parsing by 0.19 (>) and 1.09 (>>>) respectively in F1-measure over the pipeline parsing approach on the top-best parse tree. It is also not surprising to find out that the integrated parsing approach outperforms the pipeline parsing approach on 20-best parse trees by 0.67 (>>>) in F1-measure on SRL, due to its exploring a larger search space, although the integrated parsing approach integrates the SRL probability and the syntactic parsing probability in the same manner as the pipeline parsing approach on 20-best parse trees. However, the syntactic parsing performance gap between the integrated parsing approach and the pipeline parsing approach on 20-best parse trees is negligible.

Results of integrated parsing with semantic role-related features. After performing the greedy feature selection algorithm on the development data, features {SF3, SF2, sf5, sf6, SF4} as proposed in Section 4.3 are sequentially selected for syntactic parsing. As what we have assumed, knowledge about the detected semantic roles and expected semantic roles is helpful for syntactic parsing. Table 3 also lists the performance achieved with those selected features. It shows that the integration of semantic role-related features in integrated parsing significantly enhances both the performance of syntactic and semantic parsing by 0.44 (>>>) and 0.49 (>>) respectively in F1-measure. In addition, it shows that it outperforms the widely-used pipeline parsing approach on top-best parse tree by 0.63 (>>>) and 1.58 (>>>) in F1-measure on syntactic and semantic parsing, respectively. Finally, it shows that it outperforms the widely-used pipeline parsing approach on 20-best parse trees by 0.46 (>>>) and 1.16 (>>>) in F1-measure on syntactic and semantic parsing, respectively. This is very encouraging, considering the notorious difficulty and complexity of both the syntactic and semantic parsing tasks.

Table 3 also shows that our proposed method works well for both verbal SRL and nominal SRL. In addition, it shows that the performance of predicate recognition is very stable due to its high dependence on POS tagging results, rather than syntactic parsing results. Finally, it is not surprising to find out that the performance of predicate recognition when mixing verbal and nominal predicates is better than the performance of either verbal predicates or nominal predicates.

5.3 Extending the Word-based Syntactic Parser to a Character-based Syntactic Parser

The above experimental results on a word-based syntactic parser (assuming correct word segmentation) show that both syntactic and semantic parsing benefit from our integrated parsing approach. However, observing the great challenge of word segmentation in Chinese informa-

tion processing, it is still unclear whether and how much joint learning benefits character-based syntactic and semantic parsing. In this section, we extended the Ratnaparkhi parser (1999) to a character-based parser (with automatic word segmentation), and then examined the effectiveness of joint learning.

Given the three-pass process in the word-based syntactic parser, it is easy to extend it to a character-based parser for Chinese texts. This can be done by only replacing the TAG procedure in the first pass with a POSCHUNK procedure, which integrates Chinese word segmentation and POS tagging in one step, following the method described in (Ng and Low 2004). Here, each character is annotated with both a boundary tag and a POS tag. The 4 possible boundary tags include "B" for a character that begins a word and is followed by another character, "M" for a character that occurs in the middle of a word, "E" for a character that ends a word, and "S" for a character that occurs as a single-character word. For example, "北京市/Beijing city/NR" would be decomposed into three units: "北/north/B_NR", "京/capital/M_NR", and "市/city/E_NR". Also, "是/is/VC" would turn into "是/is/S_VC". Through POSCHUNK, all characters in a sentence are first assigned with POS chunk labels which must be compatible with previous ones, and then merged into words with their POS tags. For example, "北/north/B_NR", "京/capital/M_NR", and "市/city/E_NR" will be merged as "北京市/Beijing/NR", "是/is/S_VC" will become "是/is/VC". Finally the merged results of the POSCHUNK are fed into the CHUNK procedure of the second pass.

Using the same data split as the previous experiments, word segmentation achieves performance of 96.3 in F1-measure on the test data. Table 4 lists the syntactic and semantic parsing performance by adopting the character-based parser.

Table 4 shows that integrated parsing benefits syntactic and semantic parsing when automatic word segmentation is considered. However, the improvements are smaller due to the extra noise caused by automatic word segmentation. For example, our experiments show that the performance of predicate recognition drops from 93.2 to 90.3 in F1-measure when replacing correct word segmentations with automatic ones.

Method	Task	R (%)	P (%)	F1
Pipeline on top-best parse tree	Syntactic	82.23	84.28	83.24
	SRL	60.40	62.75	61.55
Pipeline on 20-best parse trees	Syntactic	82.25	84.29	83.26
	SRL	60.17	63.63	61.85
Integrated parsing with semantic role-related features	Syntactic	82.51	84.31	83.40
	SRL	60.09	65.35	62.61

Table 4: Performance with the character-based parser[1] (using automatically recognized word boundaries).

6 Conclusion

In this paper, we explore joint syntactic and semantic parsing to improve the performance of both syntactic and semantic parsing, in particular that of semantic parsing. Evaluation shows that our integrated parsing approach outperforms the pipeline parsing approach on N-best parse trees, a natural extension of the widely-used pipeline parsing approach on the top-best parse tree. It also shows that incorporating semantic information into syntactic parsing significantly improves the performance of both syntactic and semantic parsing. This is very promising and encouraging, considering the complexity of both syntactic and semantic parsing.

To our best knowledge, this is the first successful research on exploring syntactic parsing and semantic role labeling for verbal and nominal predicates in an integrated way.

Acknowledgments

The first two authors were financially supported by Projects 60683150, 60970056, and 90920004 under the National Natural Science Foundation of China. This research was also partially supported by a research grant R-252-000-225-112 from National University of Singapore Academic Research Fund. We also want to thank the reviewers for insightful comments.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL 1998*.

Xavier Carreras and Lluis Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL 2004*.

[1] POS tags are included in evaluating the performance of a character-based syntactic parser. Thus it cannot be directly compared with the word-based one where correct word segmentation is assumed.

Xavier Carreras and Lluis Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL 2005*.

Eugene Charniak. 2001. Immediate-Head Parsing for Language Models. In *Proceedings of ACL 2001*.

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint Parsing and Named Entity Recognition. In *Proceedings of NAACL 2009*.

Jan Hajic, Massimiliano Ciaramita, Richard Johansson, et al. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of CoNLL 2009*.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. In *Proceedings of EMNLP 2006*.

Fang Kong, Guodong Zhou, and Qiaoming Zhu. 2009. Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective. In *Proceedings of EMNLP 2009*.

Peter Koomen, Vasin Punyakanok, Dan Roth, Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of CoNLL 2005*.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving Nominal SRL in Chinese Language with Verbal SRL information and Automatic Predicate Recognition. In *Proceedings of EMNLP 2009*.

Chang Liu and Hwee Tou Ng. 2007. Learning Predictive Structures for Semantic Role Labeling of NomBank. In *Proceedings of ACL 2007*.

Paola Merlo and Gabriele Mussillo. 2005. Accurate Function Parsing. In *Proceedings of EMNLP 2005*.

Paola Merlo and Gabriele Musillo. 2008. Semantic Parsing for High-Precision Semantic Role Labelling. In *Proceedings of CoNLL 2008*.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating Noun Argument Structure for NomBank. In *Proceedings of LREC 2004*.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of ANLP 2000*.

Srini Narayanan and Sanda Harabagiu. 2004. Question Answering based on Semantic Structures. In *Proceedings of COLING 2004*.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese Part-of-Speech Tagging: One-at-a-Time or All-at-Once? Word-Based or Character-Based? In *Proceedings of EMNLP 2004*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31, 71-106.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceesings of NAACL 2007*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support Vector Learning for Semantic Argument Classification. *Machine Learning*, 2005, 60:11-39.

Adwait Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning*, 34, 151-175.

Mihai Surdeanu, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of ACL 2003*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluis Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of CoNLL 2008*.

Charles Sutton and Andrew McCallum. 2005. Joint Parsing and Semantic Role Labeling. In *Proceedings of CoNLL2005*.

Nianwen Xue and Martha Palmer. 2003. Annotating the Propositions in the Penn Chinese TreeBank. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing.*

Nianwen Xue. 2006. Annotating the Predicate-Argument Structure of Chinese Nominalizations. In *Proceedings of LREC 2006*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2):225-255.

Szu-ting Yi and Martha Palmer. 2005. The Integration of Syntactic Parsing and Semantic Role Labeling. In *Proceedings of CoNLL 2005*.

Yue Zhang and Stephen Clark. 2008. Joint Word Segmentation and POS Tagging Using a Single Perceptron. In *Proceedings of ACL 2008*.

Cross-Language Text Classification using Structural Correspondence Learning

Peter Prettenhofer and **Benno Stein**

Bauhaus-Universität Weimar
D-99421 Weimar, Germany
{peter.prettenhofer,benno.stein}@uni-weimar.de

Abstract

We present a new approach to cross-language text classification that builds on structural correspondence learning, a recently proposed theory for domain adaptation. The approach uses unlabeled documents, along with a simple word translation oracle, in order to induce task-specific, cross-lingual word correspondences. We report on analyses that reveal quantitative insights about the use of unlabeled data and the complexity of inter-language correspondence modeling.

We conduct experiments in the field of cross-language sentiment classification, employing English as source language, and German, French, and Japanese as target languages. The results are convincing; they demonstrate both the robustness and the competitiveness of the presented ideas.

1 Introduction

This paper deals with cross-language text classification problems. The solution of such problems requires the transfer of classification knowledge between two languages. Stated precisely: We are given a text classification task γ in a target language $\mathcal{T}$ for which no labeled documents are available. γ may be a spam filtering task, a topic categorization task, or a sentiment classification task. In addition, we are given *labeled* documents for the identical task in a different source language $\mathcal{S}$.

Such type of cross-language text classification problems are addressed by constructing a classifier $f_{\mathcal{S}}$ with training documents written in $\mathcal{S}$ and by applying $f_{\mathcal{S}}$ to unlabeled documents written in $\mathcal{T}$. For the application of $f_{\mathcal{S}}$ under language $\mathcal{T}$ different approaches are current practice: machine translation of unlabeled documents from $\mathcal{T}$ to $\mathcal{S}$, dictionary-based translation of unlabeled documents from $\mathcal{T}$ to $\mathcal{S}$, or language-independent concept modeling by means of comparable corpora. The mentioned approaches have their pros and cons, some of which are discussed below.

Here we propose a different approach to cross-language text classification which adopts ideas from the field of multi-task learning (Ando and Zhang, 2005a). Our approach builds upon structural correspondence learning, SCL, a recently proposed theory for domain adaptation in the field of natural language processing (Blitzer et al., 2006).

Similar to SCL, our approach induces correspondences among the words from both languages by means of a small number of so-called *pivots*. In our context a pivot is a pair of words, $\{w_{\mathcal{S}}, w_{\mathcal{T}}\}$, from the source language $\mathcal{S}$ and the target language $\mathcal{T}$, which possess a similar semantics. Testing the occurrence of $w_{\mathcal{S}}$ or $w_{\mathcal{T}}$ in a set of unlabeled documents from $\mathcal{S}$ and $\mathcal{T}$ yields two equivalence classes *across* these languages: one class contains the documents where either $w_{\mathcal{S}}$ or $w_{\mathcal{T}}$ occur, the other class contains the documents where neither $w_{\mathcal{S}}$ nor $w_{\mathcal{T}}$ occur. Ideally, a pivot splits the set of unlabeled documents with respect to the semantics that is associated with $\{w_{\mathcal{S}}, w_{\mathcal{T}}\}$. The correlation between $w_{\mathcal{S}}$ or $w_{\mathcal{T}}$ and other words w, $w \notin \{w_{\mathcal{S}}, w_{\mathcal{T}}\}$ is modeled by a linear classifier, which then is used as a language-independent predictor for the two equivalence classes. As we will see, a small number of pivots can capture a sufficiently large part of the correspondences between $\mathcal{S}$ and $\mathcal{T}$ in order to (1) construct a cross-lingual representation and (2) learn a classifier $f_{\mathcal{ST}}$ for the task γ that operates on this representation. Several advantages follow from our approach:

- Task specificity. The approach exploits the words' pragmatics since it considers—during the pivot selection step—task-specific characteristics of language use.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1118–1127,
Uppsala, Sweden, 11-16 July 2010.

- Efficiency in terms of linguistic resources. The approach uses unlabeled documents from both languages along with a small number (100 - 500) of translated words, instead of employing a parallel corpus or an extensive bilingual dictionary.

- Efficiency in terms of computing resources. The approach solves the classification problem directly, instead of resorting to a more general and potentially much harder problem such as machine translation. Note that the use of such technology is prohibited in certain situations (market competitors) or restricted by environmental constraints (offline situations, high latency, bandwidth capacity).

Contributions Our contributions to the outlined field are threefold: First, the identification and utilization of the theory of SCL to cross-language text classification, which has, to the best of our knowledge, not been investigated before. Second, the further development and adaptation of SCL towards a technology that is competitive with the state-of-the-art in cross-language text classification. Third, an in-depth analysis with respect to important hyperparameters such as the ratio of labeled and unlabeled documents, the number of pivots, and the optimum dimensionality of the cross-lingual representation. In this connection we compile extensive corpora in the languages English, German, French, and Japanese, and for different sentiment classification tasks.

The paper is organized as follows: Section 2 surveys related work. Section 3 states the terminology for cross-language text classification. Section 4 describes our main contribution, a new approach to cross-language text classification based on structural correspondence learning. Section 5 presents experimental results in the context of cross-language sentiment classification.

2 Related Work

Cross-Language Text Classification Bel et al. (2003) belong to the first who explicitly considered the problem of cross-language text classification. Their research, however, is predated by work in cross-language information retrieval, CLIR, where similar problems are addressed (Oard, 1998). Traditional approaches to cross-language text classification and CLIR use linguistic resources such as bilingual dictionaries or parallel corpora to induce correspondences between two languages (Lavrenko et al., 2002; Olsson et al., 2005). Dumais et al. (1997) is considered as seminal work in CLIR: they propose a method which induces semantic correspondences between two languages by performing latent semantic analysis, LSA, on a parallel corpus. Li and Taylor (2007) improve upon this method by employing kernel canonical correlation analysis, CCA, instead of LSA. The major limitation of these approaches is their computational complexity and, in particular, the dependence on a parallel corpus, which is hard to obtain—especially for less resource-rich languages. Gliozzo and Strapparava (2005) circumvent the dependence on a parallel corpus by using so-called multilingual domain models, which can be acquired from comparable corpora in an unsupervised manner. In (Gliozzo and Strapparava, 2006) they show for particular tasks that their approach can achieve a performance close to that of monolingual text classification.

Recent work in cross-language text classification focuses on the use of automatic machine translation technology. Most of these methods involve two steps: (1) translation of the documents into the source or the target language, and (2) dimensionality reduction or semi-supervised learning to reduce the noise introduced by the machine translation. Methods which follow this two-step approach include the EM-based approach by Rigutini et al. (2005), the CCA approach by Fortuna and Shawe-Taylor (2005), the information bottleneck approach by Ling et al. (2008), and the co-training approach by Wan (2009).

Domain Adaptation Domain adaptation refers to the problem of adapting a statistical classifier trained on data from one (or more) source domains (e.g., newswire texts) to a different target domain (e.g., legal texts). In the basic domain adaptation setting we are given labeled data from the source domain and unlabeled data from the target domain, and the goal is to train a classifier for the target domain. Beyond this setting one can further distinguish whether a small amount of labeled data from the target domain is available (Daume, 2007; Finkel and Manning, 2009) or not (Blitzer et al., 2006; Jiang and Zhai, 2007). The latter setting is referred to as unsupervised domain adaptation.

Note that, cross-language text classification can be cast as an unsupervised domain adaptation problem by considering each language as a separate domain. Blitzer et al. (2006) propose an effective algorithm for unsupervised domain adaptation, called structural correspondence learning. First, SCL identifies features that generalize across domains, which the authors call pivots. SCL then models the correlation between the pivots and all other features by training linear classifiers on the unlabeled data from both domains. This information is used to induce correspondences among features from the different domains and to learn a shared representation that is meaningful across both domains. SCL is related to the structural learning paradigm introduced by Ando and Zhang (2005a). The basic idea of structural learning is to constrain the hypothesis space of a learning task by considering multiple different but related tasks on the same input space. Ando and Zhang (2005b) present a semi-supervised learning method based on this paradigm, which generates related tasks from unlabeled data. Quattoni et al. (2007) apply structural learning to image classification in settings where little labeled data is given.

3 Cross-Language Text Classification

This section introduces basic models and terminology.

In standard text classification, a document d is represented under the bag-of-words model as $|V|$-dimensional feature vector $\mathbf{x} \in X$, where V, the vocabulary, denotes an ordered set of words, $x_i \in \mathbf{x}$ denotes the normalized frequency of word i in d, and X is an inner product space. $D_{\mathcal{S}}$ denotes the training set and comprises tuples of the form $(\mathbf{x}, y)$, which associate a feature vector $\mathbf{x} \in X$ with a class label $y \in Y$. The goal is to find a classifier $f : X \rightarrow Y$ that predicts the labels of new, previously unseen documents. Without loss of generality we restrict ourselves to binary classification problems and linear classifiers, i.e., $Y = \{+1, -1\}$ and $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x})$. $\mathbf{w}$ is a weight vector that parameterizes the classifier, $[\cdot]^T$ denotes the matrix transpose. The computation of $\mathbf{w}$ from $D_{\mathcal{S}}$ is referred to as model estimation or training. A common choice for $\mathbf{w}$ is given by a vector $\mathbf{w}^*$ that minimizes the regularized training error:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbf{R}^{|V|}}{\operatorname{argmin}} \sum_{(\mathbf{x},y)\in D_{\mathcal{S}}} L(y,\, \mathbf{w}^T\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \quad (1)$$

L is a loss function that measures the quality of the classifier, λ is a non-negative regularization parameter that penalizes model complexity, and $\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$. Different choices for L entail different classifier types; e.g., when choosing the hinge loss function for L one obtains the popular Support Vector Machine classifier (Zhang, 2004).

Standard text classification distinguishes between labeled (training) documents and unlabeled (test) documents. Cross-language text classification poses an extra constraint in that training documents and test documents are written in different languages. Here, the language of the training documents is referred to as source language $\mathcal{S}$, and the language of the test documents is referred to as target language $\mathcal{T}$. The vocabulary V divides into $V_{\mathcal{S}}$ and $V_{\mathcal{T}}$, called vocabulary of the source language and vocabulary of the target language, with $V_{\mathcal{S}} \cap V_{\mathcal{T}} = \emptyset$. I.e., documents from the training set and the test set map on two non-overlapping regions of the feature space. Thus, a linear classifier $f_{\mathcal{S}}$ trained on $D_{\mathcal{S}}$ associates non-zero weights only with words from $V_{\mathcal{S}}$, which in turn means that $f_{\mathcal{S}}$ cannot be used to classify documents written in $\mathcal{T}$.

One way to overcome this "feature barrier" is to find a cross-lingual representation for documents written in $\mathcal{S}$ and $\mathcal{T}$, which enables the transfer of classification knowledge between the two languages. Intuitively, one can understand such a cross-lingual representation as a concept space that underlies both languages. In the following, we will use θ to denote a map that associates the original $|V|$-dimensional representation of a document d written in $\mathcal{S}$ or $\mathcal{T}$ with its cross-lingual representation. Once such a mapping is found the cross-language text classification problem reduces to a standard classification problem in the cross-lingual space. Note that the existing methods for cross-language text classification can be characterized by the way θ is constructed. For instance, cross-language latent semantic indexing (Dumais et al., 1997) and cross-language explicit semantic analysis (Potthast et al., 2008) estimate θ using a parallel corpus. Other methods use linguistic resources such as a bilingual dictionary to obtain θ (Bel et al., 2003; Olsson et al., 2005).

4 Cross-Language Structural Correspondence Learning

We now present a novel method for learning a map θ by exploiting relations from unlabeled documents written in $\mathcal{S}$ and $\mathcal{T}$. The proposed method, which we call cross-language structural correspondence learning, CL-SCL, addresses the following learning setup (see also Figure 1):

- Given a set of labeled training documents $D_{\mathcal{S}}$ written in language $\mathcal{S}$, the goal is to create a text classifier for documents written in a different language $\mathcal{T}$. We refer to this classification task as the *target task*. An example for the target task is the determination of sentiment polarity, either positive or negative, of book reviews written in German ($\mathcal{T}$) given a set of training reviews written in English ($\mathcal{S}$).

- In addition to the labeled training documents $D_{\mathcal{S}}$ we have access to unlabeled documents $D_{\mathcal{S},u}$ and $D_{\mathcal{T},u}$ from both languages $\mathcal{S}$ and $\mathcal{T}$. Let D_u denote $D_{\mathcal{S},u} \cup D_{\mathcal{T},u}$.

- Finally, we are given a budget of calls to a word translation oracle (e.g., a domain expert) to map words in the source vocabulary $V_{\mathcal{S}}$ to their corresponding translations in the target vocabulary $V_{\mathcal{T}}$. For simplicity and without loss of applicability we assume here that the word translation oracle maps each word in $V_{\mathcal{S}}$ to exactly one word in $V_{\mathcal{T}}$.

CL-SCL comprises three steps: In the first step, CL-SCL selects word pairs $\{w_{\mathcal{S}}, w_{\mathcal{T}}\}$, called pivots, where $w_{\mathcal{S}} \in V_{\mathcal{S}}$ and $w_{\mathcal{T}} \in V_{\mathcal{T}}$. Pivots have to satisfy the following conditions:

Confidence Both words, $w_{\mathcal{S}}$ and $w_{\mathcal{T}}$, are predictive for the target task.

Support Both words, $w_{\mathcal{S}}$ and $w_{\mathcal{T}}$, occur frequently in $D_{\mathcal{S},u}$ and $D_{\mathcal{T},u}$ respectively.

The confidence condition ensures that, in the second step of CL-SCL, only those correlations are modeled that are useful for discriminative learning. The support condition, on the other hand, ensures that these correlations can be estimated accurately. Considering our sentiment classification example, the word pair $\{\text{excellent}_{\mathcal{S}}, \text{exzellent}_{\mathcal{T}}\}$ satisfies both conditions: (1) the words are strong indicators of positive sentiment, and (2) the words occur frequently in book reviews from both languages. Note that the support of $w_{\mathcal{S}}$ and $w_{\mathcal{T}}$ can be determined from the unlabeled data D_u. The confidence, however, can only be determined for $w_{\mathcal{S}}$ since the setting gives us access to labeled data from $\mathcal{S}$ only.

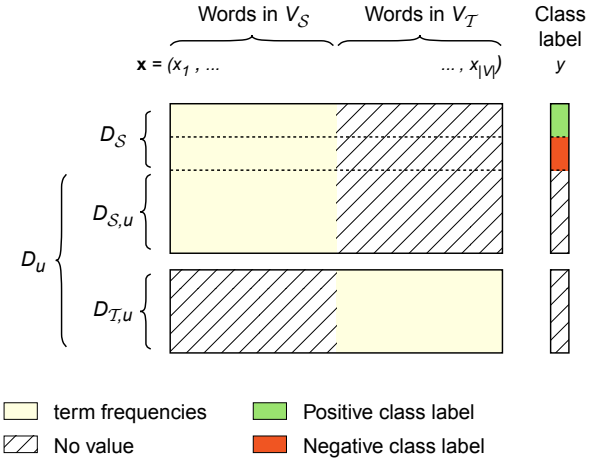


Figure 1: The document sets underlying CL-SCL. The subscripts $_{\mathcal{S}}$, $_{\mathcal{T}}$, and $_u$ designate "source language", "target language", and "unlabeled".

We use the following heuristic to form an ordered set P of pivots: First, we choose a subset V_P from the source vocabulary $V_{\mathcal{S}}$, $|V_P| \ll |V_{\mathcal{S}}|$, which contains those words with the highest mutual information with respect to the class label of the target task in $D_{\mathcal{S}}$. Second, for each word $w_{\mathcal{S}} \in V_P$ we find its translation in the target vocabulary $V_{\mathcal{T}}$ by querying the translation oracle; we refer to the resulting set of word pairs as the candidate pivots, P':

$$P' = \{\{w_{\mathcal{S}}, \textsc{translate}(w_{\mathcal{S}})\} \mid w_{\mathcal{S}} \in V_P\}$$

We then enforce the support condition by eliminating in P' all candidate pivots $\{w_{\mathcal{S}}, w_{\mathcal{T}}\}$ where the document frequency of $w_{\mathcal{S}}$ in $D_{\mathcal{S},u}$ or of $w_{\mathcal{T}}$ in $D_{\mathcal{T},u}$ is smaller than some threshold ϕ:

$$P = \textsc{candidateElimination}(P', \phi)$$

Let m denote $|P|$, the number of pivots.

In the second step, CL-SCL models the correlations between each pivot $\{w_{\mathcal{S}}, w_{\mathcal{T}}\} \in P$ and all other words $w \in V \setminus \{w_{\mathcal{S}}, w_{\mathcal{T}}\}$. This is done by training linear classifiers that predict whether or not $w_{\mathcal{S}}$ or $w_{\mathcal{T}}$ occur in a document, based on the other words. For this purpose a training set D_l is created for each pivot $p_l \in P$:

$$D_l = \{(\textsc{mask}(\mathbf{x}, p_l), \textsc{in}(\mathbf{x}, p_l)) \mid \mathbf{x} \in D_u\}$$

MASK$(\mathbf{x}, p_l)$ is a function that returns a copy of $\mathbf{x}$ where the components associated with the two words in p_l are set to zero—which is equivalent to removing these words from the feature space. IN$(\mathbf{x}, p_l)$ returns +1 if one of the components of $\mathbf{x}$ associated with the words in p_l is non-zero and -1 otherwise. For each D_l a linear classifier, characterized by the parameter vector $\mathbf{w}_l$, is trained by minimizing Equation (1) on D_l. Note that each training set D_l contains documents from both languages. Thus, for a pivot $p_l = \{w_\mathcal{S}, w_\mathcal{T}\}$ the vector $\mathbf{w}_l$ captures both the correlation between $w_\mathcal{S}$ and $V_\mathcal{S} \setminus \{w_\mathcal{S}\}$ and the correlation between $w_\mathcal{T}$ and $V_\mathcal{T} \setminus \{w_\mathcal{T}\}$.

In the third step, CL-SCL identifies correlations *across* pivots by computing the singular value decomposition of the $|V| \times m$-dimensional parameter matrix $\mathbf{W}$, $\mathbf{W} = \begin{bmatrix}\mathbf{w}_1 & \dots & \mathbf{w}_m\end{bmatrix}$:

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \text{SVD}(\mathbf{W})$$

Recall that $\mathbf{W}$ encodes the correlation structure between pivot and non-pivot words in the form of multiple linear classifiers. Thus, the columns of $\mathbf{U}$ identify common substructures among these classifiers. Choosing the columns of $\mathbf{U}$ associated with the largest singular values yields those substructures that capture most of the correlation in $\mathbf{W}$. We define θ as those columns of $\mathbf{U}$ that are associated with the k largest singular values:

$$\theta = \mathbf{U}^T_{[1:k,\, 1:|V|]}$$

Algorithm 1 summarizes the three steps of CL-SCL. At training and test time, we apply the projection θ to each input instance $\mathbf{x}$. The vector $\mathbf{v}^*$ that minimizes the regularized training error for $D_\mathcal{S}$ in the projected space is defined as follows:

$$\mathbf{v}^* = \operatorname*{argmin}_{\mathbf{v} \in \mathbf{R}^k} \sum_{(\mathbf{x},y) \in D_\mathcal{S}} L(y,\, \mathbf{v}^T\theta\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{v}\|^2 \quad (2)$$

The resulting classifier $f_{\mathcal{ST}}$, which will operate in the cross-lingual setting, is defined as follows:

$$f_{\mathcal{ST}}(\mathbf{x}) = sign(\mathbf{v}^{*T}\theta\mathbf{x})$$

4.1 An Alternative View of CL-SCL

An alternative view of cross-language structural correspondence learning is provided by the framework of structural learning (Ando and Zhang, 2005a). The basic idea of structural learning is to constrain the hypothesis space, i.e., the space of possible weight vectors, of the target task by considering multiple different but related prediction tasks. In our context these auxiliary tasks are represented by the pivot predictors, i.e., the columns of $\mathbf{W}$. Each column vector $\mathbf{w}_l$ can be considered as a linear classifier which performs well in both languages. I.e., we regard the column space of $\mathbf{W}$ as an approximation to the *subspace of bilingual classifiers*. By computing SVD$(\mathbf{W})$ one obtains a compact representation of this column space in the form of an orthonormal basis θ^T.

The subspace is used to constrain the learning of the target task by restricting the weight vector $\mathbf{w}$ to lie in the subspace defined by θ^T. Following Ando and Zhang (2005a) and Quattoni et al. (2007) we choose $\mathbf{w}$ for the target task to be $\mathbf{w}^* = \theta^T\mathbf{v}^*$, where $\mathbf{v}^*$ is defined as follows:

$$\mathbf{v}^* = \operatorname*{argmin}_{\mathbf{v} \in \mathbf{R}^k} \sum_{(\mathbf{x},y) \in D_\mathcal{S}} L(y,\, (\theta^T\mathbf{v})^T\mathbf{x}) + \frac{\lambda}{2}\|\mathbf{v}\|^2 \quad (3)$$

Since $(\theta^T\mathbf{v})^T = \mathbf{v}^T\theta$ it follows that this view of CL-SCL corresponds to the induction of a new feature space given by Equation 2.

Algorithm 1 CL-SCL

Input: Labeled source data $D_\mathcal{S}$
Unlabeled data $D_u = D_{\mathcal{S},u} \cup D_{\mathcal{T},u}$

Parameters: m, k, λ, and ϕ

Output: $k \times |V|$-dimensional matrix θ

1. SELECTPIVOTS$(D_\mathcal{S}, m)$
$V_P = \text{MUTUALINFORMATION}(D_\mathcal{S})$
$P' = \{\{w_\mathcal{S}, \text{TRANSLATE}(w_\mathcal{S})\} \mid w_\mathcal{S} \in V_P\}$
$P = \text{CANDIDATEELIMINATION}(P', \phi)$

2. TRAINPIVOTPREDICTORS(D_u, P)
for $l = 1$ **to** m **do**
$D_l = \{(\text{MASK}(\mathbf{x}, p_l),\ \text{IN}(\mathbf{x}, p_l)) \mid \mathbf{x} \in D_u\}$
$\mathbf{w}_l = \operatorname*{argmin}_{\mathbf{w} \in \mathbf{R}^{|V|}} \sum_{(\mathbf{x},y) \in D_l} L(y, \mathbf{w}^T\mathbf{x})) + \frac{\lambda}{2}\|\mathbf{w}\|^2$
end for
$\mathbf{W} = \begin{bmatrix}\mathbf{w}_1 & \dots & \mathbf{w}_m\end{bmatrix}$

3. COMPUTESVD$(\mathbf{W}, k)$
$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \text{SVD}(\mathbf{W})$
$\theta = \mathbf{U}^T_{[1:k,\, 1:|V|]}$

output $\{\theta\}$

5 Experiments

We evaluate CL-SCL for the task of cross-language sentiment classification using English as source language and German, French, and Japanese as target languages. Special emphasis is put on corpus construction, determination of upper bounds and baselines, and a sensitivity analysis of important hyperparameters. All data described in the following is publicly available from our project website.[1]

5.1 Dataset and Preprocessing

We compiled a new dataset for cross-language sentiment classification by crawling product reviews from Amazon.{de | fr | co.jp}. The crawled part of the corpus contains more than 4 million reviews in the three languages German, French, and Japanese. The corpus is extended with English product reviews provided by Blitzer et al. (2007). Each review contains a category label, a title, the review text, and a rating of 1-5 stars. Following Blitzer et al. (2007) a review with >3 (<3) stars is labeled as positive (negative); other reviews are discarded. For each language the labeled reviews are grouped according to their category label, whereas we restrict our experiments to three categories: books, dvds, and music.

Since most of the crawled reviews are positive (80%), we decide to balance the number of positive and negative reviews. In this study, we are interested in whether the cross-lingual representation induced by CL-SCL captures the difference between positive and negative reviews; by balancing the reviews we ensure that the imbalance does not affect the learned model. Balancing is achieved by deleting reviews from the majority class uniformly at random for each language-specific category. The resulting sets are split into three disjoint, balanced sets, containing training documents, test documents, and unlabeled documents; the respective set sizes are 2,000, 2,000, and 9,000-50,000. See Table 1 for details.

For each of the nine target-language-category-combinations a text classification task is created by taking the training set of the product category in $\mathcal{S}$ and the test set of the same product category in $\mathcal{T}$. A document d is described as normalized feature vector $\mathbf{x}$ under a unigram bag-of-words document representation. The morphological analyzer MeCab is used for Japanese word segmentation.[2]

5.2 Implementation

Throughout the experiments linear classifiers are employed; they are trained by minimizing Equation (1), using a stochastic gradient descent (SGD) algorithm. In particular, the learning rate schedule from PEGASOS is adopted (Shalev-Shwartz et al., 2007), and the modified Huber loss, introduced by Zhang (2004), is chosen as loss function L.[3]

SGD receives two hyperparameters as input: the number of iterations T, and the regularization parameter λ. In our experiments T is always set to 10^6, which is about the number of iterations required for SGD to converge. For the target task, λ is determined by 3-fold cross-validation, testing for λ all values $10^{-i}, i \in [0; 6]$. For the pivot prediction task, λ is set to the small value of 10^{-5}, in order to favor model accuracy over generalizability.

The computational bottleneck of CL-SCL is the SVD of the dense parameter matrix $\mathbf{W}$. Here we follow Blitzer et al. (2006) and set the negative values in $\mathbf{W}$ to zero, which yields a sparse representation. For the SVD computation the Lanczos algorithm provided by SVDLIBC is employed.[4] We investigated an alternative approach to obtain a sparse $\mathbf{W}$ by directly enforcing sparse pivot predictors $\mathbf{w}_l$ through L1-regularization (Tsuruoka et al., 2009), but didn't pursue this strategy due to unstable results. Since SGD is sensitive to feature scaling the projection $\theta\mathbf{x}$ is post-processed as follows: (1) Each feature of the cross-lingual representation is standardized to zero mean and unit variance, where mean and variance are estimated on $D_{\mathcal{S}} \cup D_u$. (2) The cross-lingual document representations are scaled by a constant α such that $|D_{\mathcal{S}}|^{-1} \sum_{\mathbf{x} \in D_{\mathcal{S}}} \|\alpha\theta\mathbf{x}\| = 1$.

We use Google Translate as word translation oracle, which returns a single translation for each query word.[5] Though such a context free translation is suboptimum we do not sanitize the returned words to demonstrate the robustness of CL-SCL with respect to translation noise. To ensure the reproducibility of our results we cache all queries to the translation oracle.

[1] http://www.webis.de/research/corpora/webis-cls-10/

[2] http://mecab.sourceforge.net

[3] Our implementation is available at http://github.com/pprett/bolt

[4] http://tedlab.mit.edu/~dr/SVDLIBC/

[5] http://translate.google.com

$\mathcal{T}$	Category	Unlabeled data		Upper Bound		CL-MT			CL-SCL		
		$\lvert D_{\mathcal{S},u}\rvert$	$\lvert D_{\mathcal{T},u}\rvert$	μ	σ	μ	σ	Δ	μ	σ	Δ
German	books	50,000	50,000	83.79	(±0.20)	**79.68**	(±0.13)	4.11	79.50	(±0.33)	4.29
	dvd	30,000	50,000	81.78	(±0.27)	**77.92**	(±0.25)	3.86	76.92	(±0.07)	4.86
	music	25,000	50,000	82.80	(±0.13)	77.22	(±0.23)	5.58	**77.79**	(±0.02)	5.00
French	books	50,000	32,000	83.92	(±0.14)	**80.76**	(±0.34)	3.16	78.49	(±0.03)	5.43
	dvd	30,000	9,000	83.40	(±0.28)	**78.83**	(±0.19)	4.57	78.80	(±0.01)	4.60
	music	25,000	16,000	86.09	(±0.13)	75.78	(±0.65)	10.31	**77.92**	(±0.03)	8.17
Japanese	books	50,000	50,000	79.39	(±0.27)	70.22	(±0.27)	9.17	**73.09**	(±0.07)	6.30
	dvd	30,000	50,000	81.56	(±0.28)	**71.30**	(±0.28)	10.26	71.07	(±0.02)	10.49
	music	25,000	50,000	82.33	(±0.13)	72.02	(±0.29)	10.31	**75.11**	(±0.06)	7.22

Table 1: Cross-language sentiment classification results. For each task, the number of unlabeled documents from $\mathcal{S}$ and $\mathcal{T}$ is given. Accuracy scores (mean μ and standard deviation σ of 10 repetitions of SGD) on the test set of the target language $\mathcal{T}$ are reported. Δ gives the difference in accuracy to the upper bound. CL-SCL uses $m = 450$, $k = 100$, and $\phi = 30$.

5.3 Upper Bound and Baseline

To get an upper bound on the performance of a cross-language method we first consider the monolingual setting. For each target-language-category-combination a linear classifier is learned on the training set and tested on the test set. The resulting accuracy scores are referred to as upper bound; it informs us about the expected performance on the target task if training data in the target language is available.

We chose a machine translation baseline to compare CL-SCL to another cross-language method. Statistical machine translation technology offers a straightforward solution to the problem of cross-language text classification and has been used in a number of cross-language sentiment classification studies (Hiroshi et al., 2004; Bautin et al., 2008; Wan, 2009). Our baseline CL-MT works as follows: (1) learn a linear classifier on the training data, and (2) translate the test documents into the source language,[6] (3) predict the sentiment polarity of the translated test documents. Note that the baseline CL-MT does not make use of unlabeled documents.

[6] Again we use Google Translate.

5.4 Performance Results and Sensitivity

Table 1 contrasts the classification performance of CL-SCL with the upper bound and with the baseline. Observe that the upper bound does not exhibit a great variability across the three languages. The average accuracy is about 82%, which is consistent with prior work on monolingual sentiment analysis (Pang et al., 2002; Blitzer et al., 2007). The performance of CL-MT, however, differs considerably between the two European languages and Japanese: for Japanese, the average difference between the upper bound and CL-MT (9.9%) is about twice as much as for German and French (5.3%). This difference can be explained by the fact that machine translation works better for European than for Asian languages such as Japanese.

Recall that CL-SCL receives three hyperparameters as input: the number of pivots m, the dimensionality of the cross-lingual representation k,

Pivot	English		German	
	Semantics	Pragmatics	Semantics	Pragmatics
{beautiful$_{\mathcal{S}}$, schön$_{\mathcal{T}}$}	amazing, beauty, lovely	picture, pattern, poetry, photographs, paintings	schöner (more beautiful), traurig (sad)	bilder (pictures), illustriert (illustrated)
{boring$_{\mathcal{S}}$, langweilig$_{\mathcal{T}}$}	plain, asleep, dry, long	characters, pages, story	langatmig (lengthy), einfach (plain), enttäuscht (disappointed)	charaktere (characters), handlung (plot), seiten (pages)

Table 2: Semantic and pragmatic correlations identified for the two pivots {beautiful$_{\mathcal{S}}$, schön$_{\mathcal{T}}$} and {boring$_{\mathcal{S}}$, langweilig$_{\mathcal{T}}$} in English and German book reviews.

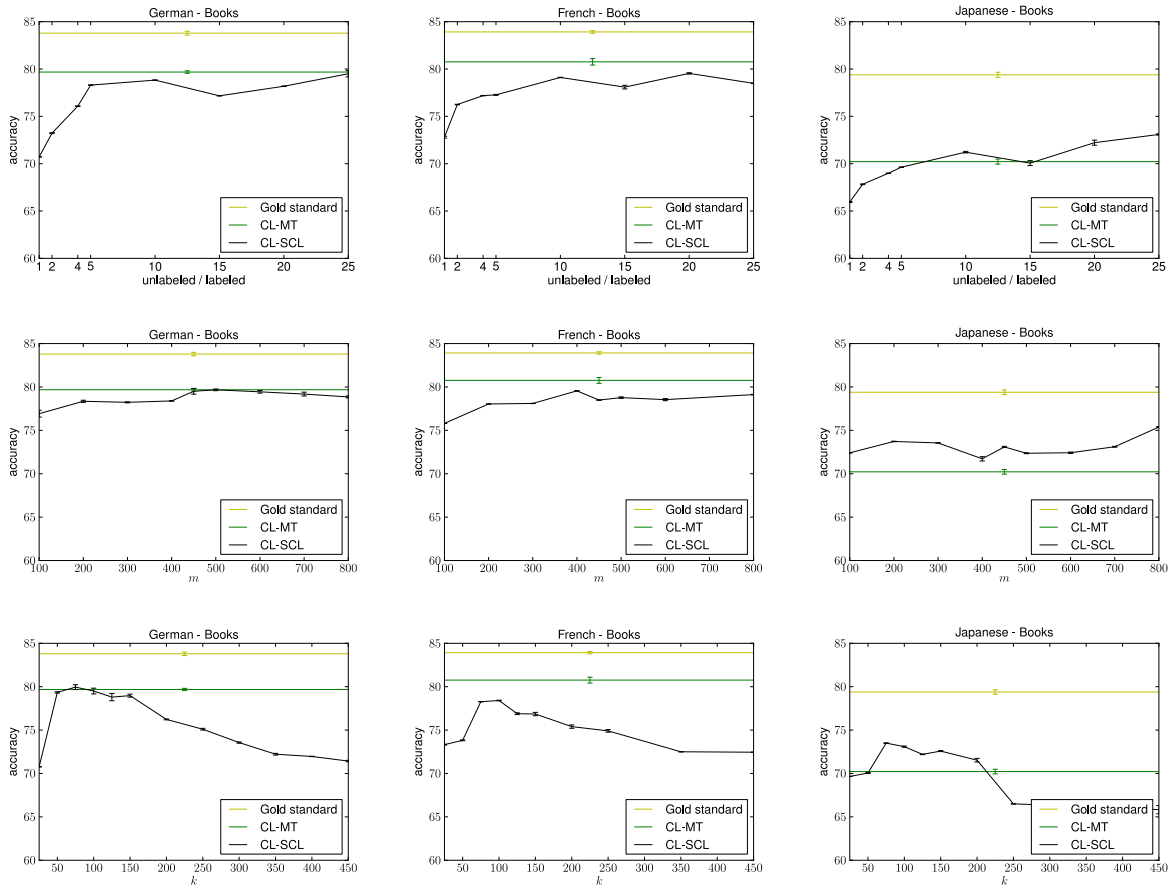


Figure 2: Influence of unlabeled data and hyperparameters on the performance of CL-SCL. The rows show the performance of CL-SCL as a function of (1) the ratio between labeled and unlabeled documents, (2) the number of pivots m, and (3) the dimensionality of the cross-lingual representation k.

and the minimum support ϕ of a pivot in $D_{\mathcal{S},u}$ and $D_{\mathcal{T},u}$. For comparison purposes we use fixed values of $m = 450$, $k = 100$, and $\phi = 30$. The results show the competitiveness of CL-SCL compared to CL-MT. Although CL-MT outperforms CL-SCL on most tasks for German and French, the difference in accuracy can be considered as small (<1%); merely for French book and music reviews the difference is about 2%. For Japanese, however, CL-SCL outperforms CL-MT on most tasks with a difference in accuracy of about 3%. The results indicate that if the difference between the upper bound and CL-MT is large, CL-SCL can circumvent the loss in accuracy. Experiments with language-specific settings revealed that for Japanese a smaller number of pivots ($150<m<250$) performs significantly better. Thus, the reported results for Japanese can be considered as pessimistic.

Primarily responsible for the effectiveness of CL-SCL is its task specificity, i.e., the ways in which context contributes to meaning (pragmatics). Due to the use of task-specific, unlabeled data, relevant characteristics are captured by the pivot classifiers. Table 2 exemplifies this with two pivots for German book reviews. The rows of the table show those words which have the highest correlation with the pivots $\{\text{beautiful}_{\mathcal{S}}, \text{schön}_{\mathcal{T}}\}$ and $\{\text{boring}_{\mathcal{S}}, \text{langweilig}_{\mathcal{T}}\}$. We can distinguish between (1) correlations that reflect similar meaning, such as “amazing”, “lovely”, or “plain”, and (2) correlations that reflect the pivot pragmatics with respect to the task, such as “picture”, “poetry”, or “pages”. Note in this connection that authors of book reviews tend to use the word “beautiful” to refer to illustrations or poetry. While the first type of word correlations can be obtained by methods that operate on parallel corpora, the second type of correlation requires an understanding of the task-specific language use.

In the following we discuss the sensitivity of each hyperparameter in isolation while keeping

the others fixed at $m = 450$, $k = 100$, and $\phi = 30$. The experiments are illustrated in Figure 2.

Unlabeled Data The first row of Figure 2 shows the performance of CL-SCL as a function of the ratio of labeled and unlabeled documents. A ratio of 1 means that $|D_{\mathcal{S},u}| = |D_{\mathcal{T},u}| = 2{,}000$, while a ratio of 25 corresponds to the setting of Table 1. As expected, an increase in unlabeled documents results in an improved performance, however, we observe a saturation at a ratio of 10 across all nine tasks.

Number of Pivots The second row shows the influence of the number of pivots m on the performance of CL-SCL. Compared to the size of the vocabularies $V_{\mathcal{S}}$ and $V_{\mathcal{T}}$, which is in 10^5 order of magnitude, the number of pivots is very small. The plots show that even a small number of pivots captures a significant amount of the correspondence between $\mathcal{S}$ and $\mathcal{T}$.

Dimensionality of the Cross-Lingual Representation The third row shows the influence of the dimensionality of the cross-lingual representation k on the performance of CL-SCL. Obviously the SVD is crucial to the success of CL-SCL if m is sufficiently large. Observe that the value of k is task-insensitive: a value of $75{<}k{<}150$ works equally well across all tasks.

6 Conclusion

The paper introduces a novel approach to cross-language text classification, called cross-language structural correspondence learning. The approach uses unlabeled documents along with a word translation oracle to automatically induce task-specific, cross-lingual correspondences. Our contributions include the adaptation of SCL for the problem of cross-language text classification and a well-founded empirical analysis. The analysis covers performance and robustness issues in the context of cross-language sentiment classification with English as source language and German, French, and Japanese as target languages. The results show that CL-SCL is competitive with state-of-the-art machine translation technology while requiring fewer resources.

Future work includes the extension of CL-SCL towards a general approach for cross-lingual adaptation of natural language processing technology.

References

Rie-K. Ando and Tong Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853.

Rie-K. Ando and Tong Zhang. 2005b. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL-05*, pages 1–9, Ann Arbor.

Mikhail Bautin, Lohit Vijayarenu, and Steven Skiena. 2008. International sentiment analysis for news and blogs. In *Proceedings of ICWSM-08*, pages 19–26, Seattle.

Nuria Bel, Cornelis H. A. Koster, and Marta Villegas. 2003. Cross-lingual text categorization. In *Proceedings of ECDL-03*, pages 126–139, Trondheim.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP-06*, pages 120–128, Sydney.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL-07*, pages 440–447, Prague.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL-07*, pages 256–263, Prague.

Susan T. Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Symposium on CrossLanguage Text and Speech Retrieval*.

Jenny-R. Finkel and Christopher-D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of HLT/NAACL-09*, pages 602–610, Boulder.

Blaž Fortuna and John Shawe-Taylor. 2005. The use of machine translation tools for cross-lingual text mining. In *Proceedings of the ICML Workshop on Learning with Multiple Views*.

Alfio Gliozzo and Carlo Strapparava. 2005. Cross language text categorization by acquiring multilingual domain models from comparable corpora. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*.

Alfio Gliozzo and Carlo Strapparava. 2006. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *Proceedings of ACL-06*, pages 553–560, Sydney.

Kanayama Hiroshi, Nasukawa Tetsuya, and Watanabe Hideo. 2004. Deeper sentiment analysis using machine translation technology. In *Proceedings of COLING-04*, pages 494–500, Geneva.

Jing Jiang and Chengxiang Zhai. 2007. A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of CIKM-07*, pages 401–410, Lisbon.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *Proceedings of SIGIR-02*, pages 175–182, Tampere.

Yaoyong Li and John S. Taylor. 2007. Advanced learning algorithms for cross-language patent retrieval and classification. *Inf. Process. Manage.*, 43(5):1183–1199.

Xiao Ling, Gui-R. Xue, Wenyuan Dai, Yun Jiang, Qiang Yang, and Yong Yu. 2008. Can chinese web pages be classified with english data source? In *Proceedings of WWW-08*, pages 969–978, Beijing.

Douglas W. Oard. 1998. A comparative study of query and document translation for cross-language information retrieval. In *Proceedings of AMTA-98*, pages 472–483, Langhorne.

J. Scott Olsson, Douglas W. Oard, and Jan Hajič. 2005. Cross-language text classification. In *Proceedings of SIGIR-05*, pages 645–646, Salvador.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP-02*, pages 79–86, Philadelphia.

Martin Potthast, Benno Stein, and Maik Anderka. 2008. A wikipedia-based multilingual retrieval model. In *Proceedings of ECIR-08*, pages 522–530, Glasgow.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. 2007. Learning visual representations using images with captions. In *Proceedings of CVPR-07*, pages 1–8, Minneapolis.

Leonardo Rigutini, Marco Maggini, and Bing Liu. 2005. An em based training algorithm for cross-language text categorization. In *Proceedings of WI-05*, pages 529–535, Compiègne.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of ICML-07*, pages 807–814, Corvalis.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of ACL/AFNLP-09*, pages 477–485, Singapore.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of ACL/AFNLP-09*, pages 235–243, Singapore.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of ICML-04*, pages 116–124, Banff.

Cross-Lingual Latent Topic Extraction

Duo Zhang
University of Illinois at Urbana-Champaign
dzhang22@cs.uiuc.edu

Qiaozhu Mei
University of Michigan
qmei@umich.edu

ChengXiang Zhai
University of Illinois at Urbana-Champaign
czhai@cs.uiuc.edu

Abstract

Probabilistic latent topic models have recently enjoyed much success in extracting and analyzing latent topics in text in an unsupervised way. One common deficiency of existing topic models, though, is that they would not work well for extracting cross-lingual latent topics simply because words in different languages generally do not co-occur with each other. In this paper, we propose a way to incorporate a bilingual dictionary into a probabilistic topic model so that we can apply topic models to extract shared latent topics in text data of different languages. Specifically, we propose a new topic model called Probabilistic Cross-Lingual Latent Semantic Analysis (PCLSA) which extends the Probabilistic Latent Semantic Analysis (PLSA) model by regularizing its likelihood function with soft constraints defined based on a bilingual dictionary. Both qualitative and quantitative experimental results show that the PCLSA model can effectively extract cross-lingual latent topics from multilingual text data.

1 Introduction

As a robust unsupervised way to perform shallow latent semantic analysis of topics in text, probabilistic topic models (Hofmann, 1999a; Blei et al., 2003b) have recently attracted much attention. The common idea behind these models is the following. A topic is represented by a multinomial word distribution so that words characterizing a topic generally have higher probabilities than other words. We can then hypothesize the existence of multiple topics in text and define a generative model based on the hypothesized topics. By fitting the model to text data, we can obtain an estimate of all the word distributions corresponding to the latent topics as well as the topic distributions in text. Intuitively, the learned word distributions capture clusters of words that co-occur with each other probabilistically.

Although many topic models have been proposed and shown to be useful (see Section 2 for more detailed discussion of related work), most of them share a common deficiency: they are designed to work only for mono-lingual text data and would not work well for extracting cross-lingual latent topics, i.e. topics shared in text data in two different natural languages. The deficiency comes from the fact that all these models rely on co-occurrences of words forming a topical cluster, but words in different language generally do not co-occur with each other. Thus with the existing models, we can only extract topics from text in each language, but cannot extract common topics shared in multiple languages.

In this paper, we propose a novel topic model, called Probabilistic Cross-Lingual Latent Semantic Analysis (PCLSA) model, which can be used to mine shared latent topics from *unaligned* text data in different languages. PCLSA extends the Probabilistic Latent Semantic Analysis (PLSA) model by regularizing its likelihood function with soft constraints defined based on a bilingual dictionary. The dictionary-based constraints are key to bridge the gap of different languages and would force the captured co-occurrences of words in each language by PCLSA to be "synchronized" so that related words in the two languages would have similar probabilities. PCLSA can be estimated efficiently using the General Expectation-Maximization (GEM) algorithm. As a topic extraction algorithm, PCLSA would take a pair of unaligned document sets in different languages and a bilingual dictionary as input, and output a set of aligned word distributions in both languages that can characterize the shared topics in the two languages. In addition, it also outputs a topic cov-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1128–1137,
Uppsala, Sweden, 11-16 July 2010.

erage distribution for each language to indicate the relative coverage of different shared topics in each language.

To the best of our knowledge, no previous work has attempted to solve this topic extraction problem and generate the same output. The closest existing work to ours is the MuTo model proposed in (Boyd-Graber and Blei, 2009) and the JointLDA model published recently in (Jagarlamudi and Daumé III, 2010). Both used a bilingual dictionary to bridge the language gap in a topic model. However, the goals of their work are different from ours in that their models mainly focus on mining cross-lingual topics of matching word pairs and discovering the correspondence at the vocabulary level. Therefore, the topics extracted using their model cannot indicate how a common topic is covered *differently* in the two languages, because the words in each word pair share the same probability in a common topic. Our work focuses on discovering correspondence at the topic level. In our model, since we only add a soft constraint on word pairs in the dictionary, their probabilities in common topics are generally different, naturally capturing which shows the different variations of a common topic in different languages.

We use a cross-lingual news data set and a review data set to evaluate PCLSA. We also propose a "cross-collection" likelihood measure to quantitatively evaluate the quality of mined topics. Experimental results show that the PCLSA model can effectively extract cross-lingual latent topics from multilingual text data, and it outperforms a baseline approach using the standard PLSA on text data in each language.

2 Related Work

Many topic models have been proposed, and the two basic models are the Probabilistic Latent Semantic Analysis (PLSA) model (Hofmann, 1999a) and the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003b). They and their extensions have been successfully applied to many problems, including hierarchical topic extraction (Hofmann, 1999b; Blei et al., 2003a; Li and McCallum, 2006), author-topic modeling (Steyvers et al., 2004), contextual topic analysis (Mei and Zhai, 2006), dynamic and correlated topic models (Blei and Lafferty, 2005; Blei and Lafferty, 2006), and opinion analysis (Mei et al., 2007; Branavan et al., 2008). Our work is an extension of PLSA by incorporating the knowledge of a bilingual dictionary as soft constraints. Such an extension is similar to the extension of PLSA for incorporating social network analysis (Mei et al., 2008a) but our constraint is different.

Some previous work on multilingual topic models assume documents in multiple languages are aligned either at the document level, sentence level or by time stamps (Mimno et al., 2009; Zhao and Xing, 2006; Kim and Khudanpur, 2004; Ni et al., 2009; Wang et al., 2007). However, in many applications, we need to mine topics from *unaligned* text corpus. For example, mining topics from search results in different languages can facilitate summarization of multilingual search results.

Besides all the multilingual topic modeling work discussed above, comparable corpora have also been studied extensively (e.g. (Fung, 1995; Franz et al., 1998; Masuichi et al., 2000; Sadat et al., 2003; Gliozzo and Strapparava, 2006)), but most previous work aims at acquiring word translation knowledge or cross-lingual text categorization from comparable corpora. Our work differs from this line of previous work in that our goal is to discover shared latent topics from multi-lingual text data that are *weakly* comparable (e.g. the data does not have to be aligned by time).

3 Problem Formulation

In general, the problem of cross-lingual topic extraction can be defined as to extract a set of common cross-lingual latent topics covered in text collections in different natural languages. A cross-lingual latent topic will be represented as a multinomial word distribution over the words in *all* the languages, i.e. a multilingual word distribution. For example, given two collections of news articles in English and Chinese, respectively, we would like to extract common topics simultaneously from the two collections. A discovered common topic, such as the terrorist attack on September 11, 2001, would be characterized by a word distribution that would assign relatively high probabilities to words related to this event in both English and Chinese (e.g. "terror", "attack", "afghanistan", "taliban", and their translations in Chinese).

As a computational problem, our input is a multi-lingual text corpus, and output is a set of cross-lingual latent topics. We now define this problem more formally.

Definition 1 (Multi-Lingual Corpus) A ***multi-lingual corpus*** $\mathcal{C}$ is a set of text collections $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s\}$, where $\mathcal{C}_i = \{d_1^i, d_2^i, \ldots, d_{M_i}^i\}$ is a collection of documents in language L_i with vocabulary $V_i = \{w_1^i, w_2^i, \ldots, w_{N_i}^i\}$. Here, M_i is the total number of documents in $\mathcal{C}_i$, N_i is the total number of words in V_i, and d_j^i is a document in collection $\mathcal{C}_i$.

Following the common assumption of bag-of-words representation, we represent document d_j^i with a bag of words $\{w_{j_1}^i, w_{j_2}^i, \ldots, w_{j_d}^i\}$, and use $c(w_k^i, d_j^i)$ to denote the count of word w_k^i in document d_j^i.

Definition 2 (Cross-Lingual Topic): A cross-lingual topic θ is a semantically coherent multinomial distribution over all the words in the vocabularies of languages L_1, ..., L_s. That is, $p(w|\theta)$ would give the probability of a word w which can be in any of the s languages under consideration. θ is semantically coherent if it assigns high probabilities to words that are semantically related either in the same language or across different languages. Clearly, we have $\sum_{i=1}^{s} \sum_{w \in V_i} p(w|\theta) = 1$ for any cross-lingual topic θ.

Definition 3 (Cross-Lingual Topic Extraction) Given a multi-lingual corpus $\mathcal{C}$, the task of cross-lingual topic extraction is to model and extract k major cross-lingual topics $\{\theta_1, \theta_2, \ldots, \theta_k\}$ from $\mathcal{C}$, where θ_i is a cross-lingual topic, and k is a user specified parameter.

The extracted cross-lingual topics can be directly used as a summary of the common content of the multi-lingual data set. Note that once a cross-lingual topic is extracted, we can easily obtain its representation in each language L_i by "splitting" the cross-lingual topic into multiple word distributions in different languages. Formally, the word distribution of a cross-lingual topic θ in language L_i is given by $p_i(w^i|\theta) = \frac{p(w^i|\theta)}{\sum_{w \in V_i} p(w|\theta)}$.

These aligned language-specific word distributions can directly review the variations of topics in different languages. They can also be used to analyze the difference of the coverage of the same topic in different languages. Moreover, they are also useful for retrieving relevant articles or passages in each language and aligning them to the same common topic, thus essentially also allowing us to integrate and align articles in multiple languages.

4 Probabilistic Cross-Lingual Latent Semantic Analysis

In this section, we present our probabilistic cross-lingual latent semantic analysis (PCLSA) model and discuss how it can be used to extract cross-lingual topics from multi-lingual text data.

The main reason why existing topic models can't be used for cross-lingual topic extraction is because they cannot cross the language barrier. Intuitively, in order to cross the language barrier and extract a common topic shared in articles in different languages, we must rely on some kind of linguistic knowledge. Our PCLSA model assumes the availability of bi-lingual dictionaries for at least some language pairs, which are generally available for major language pairs. Specifically, for text data in languages L_1, ..., L_s, if we represent each language as a node in a graph and connect those language pairs for which we have a bilingual dictionary, the minimum requirement is that the whole graph is connected. Thus, as a minimum, we will need $s-1$ distinct bilingual dictionaries. This is so that we can potentially cross all the language barriers.

Our key idea is to "synchronize" the extraction of monolingual "component topics" of a cross-lingual topic from individual languages by forcing a cross-lingual topic word distribution to assign similar probabilities to words that are potential translations according to a L_i-L_j bilingual dictionary. We achieve this by adding such preferences formally to the likelihood function of a probabilistic topic model as "soft constraints" so that when we estimate the model, we would try to not only fit the text data well (which is necessary to extract coherent component topics from each language), but also satisfy our specified preferences (which would ensure the extracted component topics in different languages are semantically related). Below we present how we implement this idea in more detail.

A bilingual dictionary for languages L_i and L_j generally would give us a many-to-many mapping between the vocabularies of the two languages. With such a mapping, we can construct a bipartite graph $G_{ij} = (V_{ij}, E_{ij})$ between the two languages where if one word can be potentially translated into another word, the two words would be connected with an edge. An edge can be weighted based on the probability of the corresponding translation. An example graph for

Chinese-English dictionary is shown in Figure 1.

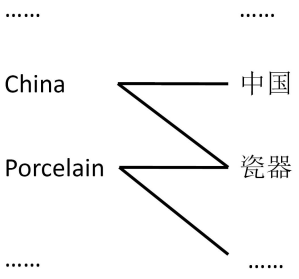


Figure 1: A Dictionary based Word Graph

With multiple bilingual dictionaries, we can merge the graphs to generate a multi-partite graph $G = (V, E)$. Based on this graph, the PCLSA model extends the standard PLSA by adding a constraint to the likelihood function to "smooth" the word distributions of topics in PLSA on the multi-partite graph so that we would encourage the words that are connected in the graph (i.e. possible translations of each other) to be given similar probabilities by every cross-lingual topic. Thus when a cross-lingual topic picks up words that co-occur in mono-lingual text, it would prefer picking up word pairs whose translations in other languages also co-occur with each other, giving us a coherent multilingual word distribution that characterizes well the content of text in different languages.

Specifically, let $\Theta = \{\theta_j\}$ $(j = 1, ..., k)$ be a set of k cross-lingual topic models to be discovered from a multilingual text data set with s languages such that $p(w|\theta_i)$ is the probability of word w according to the topic model θ_i.

If we are to use the regular PLSA to model our data, we would have the following log-likelihood and we usually use a maximum likelihood estimator to estimate parameters and discover topics.

$$L(\mathcal{C}) = \sum_{i=1}^{s} \sum_{d \in \mathcal{C}_i} \sum_{w} c(w, d) \log \sum_{j=1}^{k} p(\theta_j|d) p(w|\theta_j)$$

Our main extension is to add to $L(\mathcal{C})$ a cross-lingual constraint term $R(\mathcal{C})$ to incorporate the knowledge of bilingual dictionaries. $R(\mathcal{C})$ is defined as

$$R(\mathcal{C}) = \frac{1}{2} \sum_{\langle u,v \rangle \in E} w(u, v) \sum_{j=1}^{k} \left(\frac{p(w_u|\theta_j)}{Deg(u)} - \frac{p(w_v|\theta_j)}{Deg(v)}\right)^2$$

where $w(u, v)$ is the weight on the edge between u and v in the multi-partite graph $G = (V, E)$, which in our experiments is set to 1, and $Deg(u)$ is the degree of word u, i.e. the sum of the weights of all the edges ending with u.

Intuitively, $R(\mathcal{C})$ measures the difference between $p(w_u|\theta_j)$ and $p(w_v|\theta_j)$ for each pair (u, v) in a bilingual dictionary; the more they differ, the larger $R(\mathcal{C})$ would be. So it can be regarded as a "loss function" to help us assess how well the "component word distributions" in multiple languages are correlated semantically. Clearly, we would like the extracted topics to have a small $R(\mathcal{C})$. We choose this specific form of loss function because it would make it convenient to solve the optimization problem of maximizing the corresponding regularized maximum likelihood (Mei et al., 2008b). The normalization with $Deg(u)$ and $Deg(v)$ can be regarded as a way to compensate for the potential ambiguity of u and v in their translations.

Putting $L(\mathcal{C})$ and $R(\mathcal{C})$ together, we would like to maximize the following objective function which is a regularized log-likelihood:

$$O(\mathcal{C}, G) = (1 - \lambda) L(\mathcal{C}) - \lambda R(\mathcal{C}) \quad (1)$$

where $\lambda \in (0, 1)$ is a parameter to balance the likelihood and the regularizer. When $\lambda = 0$, we recover the standard PLSA.

Specifically, we will search for a set of values for all our parameters that can maximize the objective function defined above. Our parameters include all the cross-lingual topics and the coverage distributions of the topics in all documents, which we denote by $\Psi = \{p(w|\theta_j), p(\theta_j|d)\}_{d,w,j}$ where $j = 1, ..., k$, w varies over the entire vocabularies of all the languages , d varies over all the documents in our collection. This optimization problem can be solved using a Generalized Expectation-Maximization (GEM) algorithm as described in (Mei et al., 2008a).

Specifically, in the **E-step** of the algorithm, the distribution of hidden variables is computed using Eq. 2.

$$z(w, d, j) = \frac{p(\theta_j|d) p(w|\theta_j)}{\sum_{j'} p(\theta_{j'}|d) p(w|\theta_{j'})} \quad (2)$$

Then in the **M-step**, we need to maximize the complete data likelihood $Q(\Psi; \Psi_n)$:

$$Q(\Psi; \Psi_n) = (1 - \lambda) L'(\mathcal{C}) - \lambda R(\mathcal{C})$$

where

$$L'(\mathcal{C}) = \sum_d \sum_w c(w,d) \sum_j z(w,d,j) \log p(\theta_j|d) p(w|\theta_j), \quad (3)$$

with the constraints that $\sum_j p(\theta_j|d) = 1$ and $\sum_w p(w|\theta_j) = 1$.

There is a closed form solution if we only want to maximize the $L'(\mathcal{C})$ part:

$$\begin{aligned} p^{(n+1)}(\theta_j|d) &= \frac{\sum_w c(w,d) z(w,d,j)}{\sum_w \sum_{j'} c(w,d) z(w,d,j')} \\ p^{(n+1)}(w|\theta_j) &= \frac{\sum_d c(w,d) z(w,d,j)}{\sum_d \sum_w' c(w',d) z(w',d,j)} \end{aligned} \quad (4)$$

However, there is no closed form solution in the M-step for the whole objective function. Fortunately, according to GEM we do not need to find the local maximum of $Q(\Psi; \Psi_n)$ in every M-step, and we only need to find a new value Ψ_{n+1} to improve the complete data likelihood, i.e. to make sure $Q(\Psi_{n+1}; \Psi_n) \geq Q(\Psi_n; \Psi_n)$. So our method is to first maximize the $L'(\mathcal{C})$ part using Eq. 4 and then use Eq. 5 to gradually increase the $R(\mathcal{C})$ part.

$$\begin{aligned} p^{(t+1)}(w_u|\theta_j) &= (1-\alpha) p^{(t)}(w_u|\theta_j) \\ &+ \alpha \sum_{\langle u,v \rangle \in E} \frac{w(u,v)}{Deg(v)} p^{(t)}(w_v|\theta_j) \end{aligned} \quad (5)$$

Here, parameter α is the length of each smoothing step. Obviously, after each smoothing step, the sum of the probabilities of all the words in one topic is still equal to 1. We smooth the parameters until we cannot get a better parameter set Ψ_{n+1}. Then, we continue to the next E-step. If there is no Ψ_{n+1} s.t. $Q(\Psi_{n+1}; \Psi_n) \geq Q(\Psi_n; \Psi_n)$, then we consider Ψ_n to be the local maximum point of the objective function Eq. 1.

5 Experiment Design

5.1 Data Set

The data set we used in our experiment is collected from news articles of Xinhua English and Chinese newswires. The whole data set is quite big, containing around 40,000 articles in Chinese and 35,000 articles in English. For different purpose of our experiments, we randomly selected different number of documents from the whole corpus, and we will describe the concrete statistics in each experiment. To process the Chinese corpus, we use a simple segmenter[1] to split the data into Chinese phrases. Both Chinese and English stopwords are removed from our data.

The dictionary file we used for our PCLSA model is from mandarintools.com[2]. For each Chinese phrase, if it has several English meanings, we add an edge between it and each of its English translation. If one English translation is an English phrase, we add an edge between the Chinese phrase and each English word in the phrase.

5.2 Baseline Method

As a baseline method, we can apply the standard PLSA (Hofmann, 1999a) directly to the multilingual corpus. Since PLSA takes advantage of the word co-occurrences in the document level to find semantic topics, directly using it for a multilingual corpus will result in finding topics mainly reflecting a single language (because words in different languages would not co-occur in the same document in general). That is, the discovered topics are mostly monolingual. These monolingual topics can then be aligned based on a bilingual dictionary to suggest a possible cross-lingual topic.

6 Experimental Results

6.1 Qualitative Comparison

To qualitatively compare PCLSA with the baseline method, we compare the word distributions of topics extracted by them. The data set we used in this experiment is selected from the Xinhua News data during the period from Jun. 8th, 2001 to Jun. 15th, 2001. There are totally 1799 English articles and 1485 Chinese articles in the data set. The number of topics to be extracted is set to 10 for both methods.

Table 1 shows the experimental results. To make it easier to understand, we add an English translation to each Chinese phrase in our results. The first ten rows show sample topics of the modeling results of traditional PLSA model. We can see that it only contains mono-language topics, i.e. the topics are either in Chinese or in English. The next ten rows are the results from our PCLSA model. Compared with the baseline method, PCLSA can not only find coherent topics from the cross-lingual corpus, but it can also show the content about one topic from both two language corpora. For example, in 'Topic 2'

[1] http://www.mandarintools.com/segmenter.html
[2] http://www.mandarintools.com/cedict.html

Table 2: Synthetic Data Set from Xinhua News

English	Shrine 90	Olympic 101	Championship 70
Chinese	CPC Anniversary 95	Afghan War 206	Championship 72

which is about 'Israel' and 'Palestinian', the Chinese corpus mentions a lot about 'Arafat' who is the leader of 'Palestinian', while the English corpus discusses more on topics such as 'cease fire' and 'women'. Similarly, in 'Topic 9', the topic is related to Philippine, the Chinese corpus mentions some environmental situation in Philippine, while the English corpus mentions a lot about 'Abu Sayyaf'.

6.2 Discovering Common Topics

To demonstrate the ability of PCLSA for finding common topics in cross-lingual corpus, we use some event names, e.g. 'Shrine' and 'Olympic', as queries and randomly select a certain number of documents from the whole corpus, which are related to the queries. The number of documents for each query in the synthetic data set is shown in Table 2. In either the English corpus or the Chinese corpus, we select a smaller number of documents about topic 'Championship' combined with the other two topics in the same corpus. In this way, when we want to extract two topics from either English or Chinese corpus, the 'Championship' topic may not be easy to extract, because the other two topics have more documents in the corpus. However, when we use PCLSA to extract four topics from the two corpora together, we expect that the topic 'Championship' will be found, because now the sum of English and Chinese documents related to 'Championship' is larger than other topics. The experimental result is shown in Table 3. The first two columns are the two topics extracted from Engish corpus, the third and the forth columns are two topics from Chinese corpus, and the other four columns are the results from cross-lingual corpus. We can see that in either the Chinese sub-collection or the English sub-collection, the topic 'Championship' is not extracted as a significant topic. But, as expected, the topic 'Championship' is extracted from the cross-lingual corpus, while the topic 'Olympic' and topic 'Shrine' are merged together. This demonstrate that PCLSA is capable of extracting common topics from a cross-lingual corpus.

6.3 Quantitative Evaluation

We also quantitatively evaluate how well our PCLSA model can discover common topics among corpus in different languages. We propose a "cross-collection" likelihood measure for this purpose. The basic idea is: suppose we got k cross-lingual topics from the whole corpus, then for each topic, we split the topic into two separate set of topics, English topics and Chinese topics, using the splitting formula described before, i.e. $p_i(w^i|\theta) = \frac{p(w^i|\theta)}{\sum_{w \in V_i} p(w|\theta)}$. Then, we use the word distribution of the Chinese topics (translating the words into English) to fit the English Corpus and use the word distribution of the English topics (translating the words into Chinese) to fit the Chinese Corpus. If the topics mined are common topics in the whole corpus, then such a "cross-collection" likelihood should be larger than those topics which are not commonly shared by the English and the Chinese corpus. To calculate the likelihood of fitness, we use the folding-in method proposed in (Hofmann, 2001). To translate topics from one language to another, e.g. Chinese to English, we look up the bilingual dictionary and do word-to-word translation. If one Chinese word has several English translations, we simply distribute its probability mass equally to each English translation.

For comparison, we use the standard PLSA model as the baseline. Basically, suppose PLSA mined k semantic topics in the Chinese corpus and k semantic topics in the English corpus. Then, we also use the "cross-collection" likelihood measure to see how well those k semantic Chinese topics fit the English corpus and those k semantic English topics fit the Chinese corpus.

We totally collect three data sets to compare the performance. For the first data set, (English_1, Chinese_1), both the Chinese and English corpus are chosen from the Xinhua News Data during the period from 2001.06.08 to 2001.06.15, which has 1799 English articles and 1485 Chinese articles. For the second data set, (English_2, Chinese_2), the Chinese corpus Chinese_2 is the same as Chinese_1, but the English corpus is chosen from 2001.06.14 to 2001.06.19 which has 1547 documents. For the third data set, (English_3, Chinese_3), the Chinese corpus is the same as in data set one, but the English corpus is chosen from 2001.10.02 to 2001.10.07 which contains 1530 documents. In other words, in the first data set,

Table 1: Qualitative Evaluation

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
党(party)	犯罪(crime)	选手(athlete)	巴(palestine)	合作(collaboration)	教育(education)	israel	bt	dollar	china
共产党(communist)	农业(agriculture)	冠军(champion)	巴勒斯坦(palestine)	上海(shanghai)	球(ball)	palestinian	beat	percent	cooperate
革命(revolution)	旅游(travel)	锦标赛(championship)	以色列(israel)	关系(relation)	联赛(league)	eu	final	million	shanghai
党员(party member)	邪教(heathendom)	底(base)	停火(cease fire)	两国(bilateral)	足球(soccer)	police	championship	index	develop
中央(central)	公安(public security)	羽毛球(badminton)	联合国(UN)	贸易(trade)	分钟(minute)	report	play	stock	beije
主义(ism)	钱宏(name)	体育(sports)	中东(mid east)	总统(president)	队员(team member)	secure	champion	point	particulate
干部(cadre)	案(case)	决赛(final)	黎巴嫩(lebanon)	国(country)	教师(teacher)	kill	win	share	matter
毛泽东(chairman mao)	执法(law enforcement)	女子(women)	马其顿(macedon)	友好(friendly)	中小学(school)	europe	olympic	close	sco
中共(chinese communist)	市(city)	象棋(chess)	冲突(conflict)	会晤(meet)	球队(team)	egypt	game	0	invest
领导(leader)	处罚(penalize)	健身(fitness)	会谈(talk)	俄罗斯(russia)	甲(grade A)	treaty	cup	billion	project
两国(bilateral)	联赛(league)	israel	cooperate	选手(athlete)	party	eu	invest	0	吸入(absorb)
合作(collaboration)	钱宏(name)	以色列(israel)	sco	particulate	党(party)	khatami	投资(investment)	dollar	II
会谈(talk)	球(ball)	bt	develop	冠军	communist	ireland	亿元(billion)	percent	阿布沙耶夫(abu)
友好(friendly)	申花(shenhua)	palestinian	country	athlete	revolution	爱尔兰(ireland)	教育(education)	index	I
巴(palestine)	主场(host)	ceasefire	president	champion	主义(-ism)	elect	环保(environ. protect.)	million	颗粒(particle)
country	A	阿拉法特(arafat)	apec	ii	抗日(antiwar)	vote	资金(money)	stock	philippine
联合国(UN)	ball	women	shanghai	象棋(chess)	同志(comrade)	presidential	中小学(school)	billion	abu
领导人(leader)	金德(jinde)	jerusalem	africa	competition	革命(revolution)	cpc	market	point	底(base)
bilateral	赛季(season)	mideast	meet	contestant	党组织(party)	iran	教师(teacher)	十亿(billion)	III
state	球员(player)	lebanon	江泽民(zemin jiang)	体操(gymnastics)	ideology	referendum	business	share	物(object)

Table 3: Effectiveness of Extracting Common Topics

English 1	English 2	Chinese 1	Chinese 2	Cross 1	Cross 2	Cross 3	Cross 4
japan	olympic	共产党(CPC)	阿富汗(afghan)	koizumi	利班(taliban)	swim	工人(worker)
shrine	ioc	锦(championship)	塔(taliban)	yasukuni	军事(military)	锦(championship)	party
visit	beije	世(world)	利班(taliban)	ioc	city	自由泳(free style)	三个(three)
koizumi	game	思想(thought)	军事(military)	japan	refugee	跳水(diving)	马克思(marx)
yasukuni	july	理论(theory)	打击(attack)	olympic	side	锦标赛(championship)	communist
war	bid	马克思(marx)	美军(US army)	beije	美军(US army)	半决赛(semi final)	marx
august	swim	游泳(swim)	丹(laden)	shrine	轰炸(bomb)	competition	theory
asia	vote	锦标赛(championship)	部队(army)	visit	喀布尔(kabul)	游泳(swim)	建党(found party)
criminal	championship	党(party)	轰炸(bomb)	奥运会(olympic)	空袭(attack)	赛纪录(record)	共产党(CPC)
ii	committee	建党(found party)	喀布尔(kabul)	奥林匹克(olympic)	难民(refugee)	罗雪娟(xuejuan luo)	revolution

the English corpus and Chinese corpus are comparable with each other, because they cover similar events during the same period. In the second data set, the English and Chinese corpora share some common topics during the overlap period. The third data is the most tough one since the two corpora are from different periods. The purpose of using these three different data sets for evaluation is to test how well PCLSA can mine common topics from either a data set where the English corpus and the Chinese corpus are comparable or a data set where the English corpus and the Chinese corpus rarely share common topics.

The experimental results are shown in Table 4. Each row shows the "cross-collection" likelihood of using the "cross-collection" topics to fit the data set named in the first column. For example, in the first row, the values are the "cross-collection" likelihood of using Chinese topics found by different methods from the first data set to fit English_1. The last collum shows how much improvement we got from PCLSA compared with PLSA. From the results, we can see that in all the data sets, our PCLSA has higher "cross-collection" likelihood value, which means it can find better common topics compared to the baseline method. Notice that the Chinese corpora are the same in all three data sets. The results show that both PCLSA and PLSA get lower "cross-collection" likelihood for fitting the Chinese corpora when the data set becomes "tougher", i.e. less topic overlapping, but the improvement of PCLSA over PLSA does not drop much. On the other hand, the improvement of PCLSA over PLSA on the three English corpora does not show any correlation with the difficulty of the data set.

Table 4: Quantitative Evaluation of Common Topic Finding ("cross-collection" log-likelihood)

	PCLSA	PLSA	Rel. Imprv.
English_1	-2.86294E+06	-3.03176E+06	5.6%
Chinese_1	-4.69989E+06	-4.85369E+06	3.2%
English_2	-2.48174E+06	-2.60805E+06	4.8%
Chinese_2	-4.73218E+06	-4.88906E+06	3.2%
English_3	-2.44714E+06	-2.60540E+06	6.1%
Chinese_3	-4.79639E+06	-4.94273E+06	3.0%

6.4 Extracting from Multi-Language Corpus

In the previous experiments, we have shown the capability and effectiveness of the PCLSA model in latent topic extraction from two language corpora. In fact, the proposed model is general and capable of extracting latent topics from multi-language corpus. For example, if we have dictionaries among multiple languages, we can construct a multi-partite graph based on the correspondence between those vocabularies, and then smooth the PCLSA model with this graph.

To show the effectiveness of PCLSA in mining multiple language corpus, we first construct a simulated data set based on 1115 reviews of three brands of laptops, namely IBM (303), Apple(468) and DELL(344). To simulate a three language cor-

Table 5: Effectiveness of Latent Topic Extraction from Multi-Language Corpus

Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
cd(apple)	battery(dell)	mouse(dell)	print(apple)	port(ibm)	laptop(ibm)	os(apple)	port(dell)
port(apple)	drive(dell)	button(dell)	resolution(dell)	card(ibm)	t20(ibm)	run(apple)	2(dell)
drive(apple)	8200(dell)	touchpad(dell)	burn(apple)	modem(ibm)	thinkpad(ibm)	1(apple)	usb(dell)
airport(apple)	inspiron(dell)	pad(dell)	normal(dell)	display(ibm)	battery(ibm)	ram(apple)	1(dell)
firewire(apple)	system(dell)	keyboard(dell)	image(dell)	built(ibm)	notebook(ibm)	mac(apple)	0(dell)
dvd(apple)	hour(dell)	point(dell)	digital(apple)	swap(ibm)	ibm(ibm)	battery(apple)	slot(dell)
usb(apple)	sound(dell)	stick(dell)	organize(apple)	easy(ibm)	3(ibm)	hour(apple)	firewire(dell)
rw(apple)	dell(dell)	rest(dell)	cds(apple)	connector(ibm)	feel(ibm)	12(apple)	display(dell)
card(apple)	service(dell)	touch(dell)	latch(apple)	feature(ibm)	hour(ibm)	operate(apple)	standard(dell)
mouse(apple)	life(dell)	erase(dell)	advertise(dell)	cd(ibm)	high(ibm)	word(apple)	fast(dell)
osx(apple)	applework(apple)	port(dell)	battery(dell)	lightest(ibm)	uxga(dell)	light(ibm)	battery(apple)
memory(dell)	file(apple)	port(apple)	battery(ibm)	quality(dell)	ultrasharp(dell)	ultrabay(ibm)	point(dell)
special(dell)	bounce(apple)	port(ibm)	battery(apple)	year(ibm)	display(dell)	connector(ibm)	touchpad(dell)
crucial(dell)	quit(apple)	firewire(apple)	geforce4(dell)	hassle(ibm)	organize(apple)	dvd(ibm)	button(dell)
memory(apple)	word(apple)	imac(apple)	100mhz(apple)	bania(dell)	learn(apple)	nice(ibm)	hour(apple)
memory(ibm)	file(ibm)	firewire(dell)	440(dell)	800mhz(apple)	logo(apple)	modem(ibm)	battery(ibm)
netscape(apple)	file(dell)	firewire(ibm)	bus(apple)	trackpad(apple)	postscript(apple)	connector(dell)	battery(dell)
reseller(apple)	microsoft(apple)	jack(apple)	8200(dell)	cover(ibm)	ll(apple)	light(apple)	fan(dell)
10(dell)	ms(apple)	playback(dell)	8100(dell)	workmanship(dell)	sxga(dell)	light(dell)	erase(dell)
special(apple)	excel(apple)	jack(dell)	chipset(dell)	section(apple)	warm(apple)	floppy(ibm)	point(apple)
2000(ibm)	ram(apple)	port(dell)	itune(apple)	uxga(dell)	port(apple)	pentium(dell)	drive(ibm)
window(ibm)	ram(ibm)	port(apple)	applework(apple)	screen(dell)	port(ibm)	processor(dell)	drive(dell)
2000(apple)	ram(dell)	port(ibm)	imovie(apple)	screen(ibm)	port(dell)	p4(dell)	drive(apple)
2000(dell)	screen(apple)	2(dell)	import(apple)	screen(apple)	usb(apple)	power(dell)	hard(ibm)
window(apple)	1(apple)	2(apple)	battery(apple)	ultrasharp(dell)	plug(apple)	pentium(apple)	osx(apple)
window(dell)	screen(ibm)	2(ibm)	iphoto(apple)	1600x1200(dell)	cord(apple)	pentium(ibm)	hard(dell)
portege(ibm)	screen(dell)	speak(dell)	battery(ibm)	display(dell)	usb(ibm)	keyboard(dell)	hard(apple)
option(ibm)	1(ibm)	toshiba(dell)	battery(dell)	display(apple)	usb(dell)	processor(ibm)	card(ibm)
hassle(ibm)	1(dell)	speak(ibm)	hour(apple)	display(ibm)	firewire(apple)	processor(apple)	dvd(ibm)
device(ibm)	maco(apple)	toshiba(ibm)	hour(ibm)	view(dell)	plug(ibm)	power(apple)	card(dell)

pus, we use an 'IBM' word, an 'Apple' word, and a 'Dell' word to replace an English word in their corpus. For example, we use 'IBM10', 'Apple10', 'Dell10' to replace the word 'CD' whenever it appears in an IBM's, Apple's, or Dell's review. After the replacement, the reviews about IBM, Apple, and Dell will not share vocabularies with each other. On the other hand, for any three created words which represent the same English word, we add three edges among them, and therefore we get a simulated dictionary graph for our PCLSA model.

The experimental result is shown in Table 5, in which we try to extract 8 topics from the cross-lingual corpus. The first ten rows show the result of our PCLSA model, in which we set a very small value to the weight parameter λ for the regularizer part. This can be used as an approximation of the result from the traditional PLSA model on this three language corpus. We can see that the extracted topics are mainly written in mono-language. As we set the value of parameter λ larger, the extracted topics become multi-lingual, which is shown in the next ten rows. From this result, we can see the difference between the reviews of different brands about the similar topic. In addition, if we set the λ even larger, we will get topics that are mostly made of the same words from the three different brands, which means the extracted topics are very smooth on the dictionary graph now.

7 Conclusion

In this paper, we study the problem of cross-lingual latent topic extraction where the task is to extract a set of common latent topics from multi-lingual text data. We propose a novel probabilistic topic model (i.e. the Probabilistic Cross-Lingual Latent Semantic Analysis (PCLSA) model) that can incorporate translation knowledge in bilingual dictionaries as a regularizer to constrain the parameter estimation so that the learned topic models would be synchronized in multiple languages. We evaluated the model using several data sets. The experimental results show that PCLSA is effective in extracting common latent topics from multilingual text data, and it outperforms the baseline method which uses the standard PLSA to fit each monolingual text data set.

Our work opens up some interesting future research directions to further explore. First, in this paper, we have only experimented with uniform weighting of edge in the bilingual graph. It should be very interesting to explore how to assign weights to the edges and study whether weighted graphs can further improve performance. Second, it would also be interesting to further extend PCLSA to accommodate discovering topics in each language that aren't well-aligned with other languages.

8 Acknowledgments

We sincerely thank the anonymous reviewers for their comprehensive and constructive comments. The work was supported in part by NASA grant

NNX08AC35A, by the National Science Foundation under Grant Numbers IIS-0713581, IIS-0713571, and CNS-0834709, and by a Sloan Research Fellowship.

References

David Blei and John Lafferty. 2005. Correlated topic models. In *NIPS '05: Advances in Neural Information Processing Systems 18*.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. 2003a. Hierarchical topic models and the nested chinese restaurant process. In *Neural Information Processing Systems (NIPS) 16*.

D. Blei, A. Ng, and M. Jordan. 2003b. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

J. Boyd-Graber and D. Blei. 2009. Multilingual topic models for unaligned text. In *Uncertainty in Artificial Intelligence*.

S. R. K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. 2008. Learning document-level semantic properties from free-text annotations. In *Proceedings of ACL 2008*.

Martin Franz, J. Scott McCarley, and Salim Roukos. 1998. Ad hoc and multilingual information retrieval at IBM. In *Text REtrieval Conference*, pages 104–115.

Pascale Fung. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of ACL 1995*, pages 236–243.

Alfio Gliozzo and Carlo Strapparava. 2006. Exploiting comparable corpora and bilingual dictionaries for cross-language text categorization. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 553–560, Morristown, NJ, USA. Association for Computational Linguistics.

T. Hofmann. 1999a. Probabilistic latent semantic analysis. In *Proceedings of UAI 1999*, pages 289–296.

Thomas Hofmann. 1999b. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *IJCAI' 99*, pages 682–687.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196.

Jagadeesh Jagaralamudi and Hal Daumé III. 2010. Extracting multilingual topics from unaligned corpora. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, Milton Keynes, United Kingdom.

Woosung Kim and Sanjeev Khudanpur. 2004. Lexical triggers and latent semantic analysis for cross-lingual language model adaptation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(2):94–112.

Wei Li and Andrew McCallum. 2006. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 577–584.

H. Masuichi, R. Flournoy, S. Kaufmann, and S. Peters. 2000. A bootstrapping method for extracting bilingual text pairs. In *Proc. 18th COLINC*, pages 1066–1070.

Qiaozhu Mei and ChengXiang Zhai. 2006. A mixture model for contextual text mining. In *Proceedings of KDD '06*, pages 649–655.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of WWW '07*.

Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008a. Topic modeling with network regularization. In *WWW*, pages 101–110.

Qiaozhu Mei, Duo Zhang, and ChengXiang Zhai. 2008b. A general optimization framework for smoothing language models on graph structures. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 611–618, New York, NY, USA. ACM.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew Mccallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889, Singapore, August. Association for Computational Linguistics.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from wikipedia. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 1155–1156, New York, NY, USA. ACM.

F. Sadat, M. Yoshikawa, and S. Uemura. 2003. Bilingual terminology acquisition from comparable corpora and phrasal translation to cross-language information retrieval. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 141–144.

Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of KDD'04*, pages 306–315.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 784–793, New York, NY, USA. ACM.

Bing Zhao and Eric P. Xing. 2006. Bitam: Bilingual topic admixture models for word alignment. In *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*.

Topic Models for Word Sense Disambiguation and Token-based Idiom Detection

Linlin Li, Benjamin Roth, and Caroline Sporleder
Saarland University, Postfach 15 11 50
66041 Saarbrücken, Germany
{linlin, beroth, csporled}@coli.uni-saarland.de

Abstract

This paper presents a probabilistic model for sense disambiguation which chooses the best sense based on the conditional probability of sense paraphrases given a context. We use a topic model to decompose this conditional probability into two conditional probabilities with latent variables. We propose three different instantiations of the model for solving sense disambiguation problems with different degrees of resource availability. The proposed models are tested on three different tasks: coarse-grained word sense disambiguation, fine-grained word sense disambiguation, and detection of literal vs. nonliteral usages of potentially idiomatic expressions. In all three cases, we outperform state-of-the-art systems either quantitatively or statistically significantly.

1 Introduction

Word sense disambiguation (WSD) is the task of automatically determining the correct sense for a target word given the context in which it occurs. WSD is an important problem in NLP and an essential preprocessing step for many applications, including machine translation, question answering and information extraction. However, WSD is a difficult task, and despite the fact that it has been the focus of much research over the years, state-of-the-art systems are still often not good enough for real-world applications. One major factor that makes WSD difficult is a relative lack of manually annotated corpora, which hampers the performance of supervised systems.

To address this problem, there has been a significant amount of work on unsupervised WSD that does not require manually sense-disambiguated training data (see McCarthy (2009) for an overview). Recently, several researchers have experimented with topic models (Brody and Lapata, 2009; Boyd-Graber et al., 2007; Boyd-Graber and Blei, 2007; Cai et al., 2007) for sense disambiguation and induction. Topic models are generative probabilistic models of text corpora in which each document is modelled as a mixture over (latent) topics, which are in turn represented by a distribution over words.

Previous approaches using topic models for sense disambiguation either embed topic features in a supervised model (Cai et al., 2007) or rely heavily on the structure of hierarchical lexicons such as WordNet (Boyd-Graber et al., 2007). In this paper, we propose a novel framework which is fairly resource-poor in that it requires only 1) a large unlabelled corpus from which to estimate the topics distributions, and 2) paraphrases for the possible target senses. The paraphrases can be user-supplied or can be taken from existing resources.

We approach the sense disambiguation task by choosing the best sense based on the conditional probability of sense paraphrases given a context. We propose three models which are suitable for different situations: Model I requires knowledge of the prior distribution over senses and directly maximizes the conditional probability of a sense given the context; Model II maximizes this conditional probability by maximizing the cosine value of two topic-document vectors (one for the sense and one for the context). We apply these models to coarse- and fine-grained WSD and find that they outperform comparable systems for both tasks.

We also test our framework on the related task of idiom detection, which involves distinguishing literal and nonliteral usages of potentially ambiguous expressions such as *rock the boat*. For this task, we propose a third model. Model III calculates the probability of a sense given a context according to the component words of the sense

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1138–1147,
Uppsala, Sweden, 11-16 July 2010.

paraphrase. Specifically, it chooses the sense type which maximizes the probability (given the context) of the paraphrase component word with the highest likelihood of occurring in that context. This model also outperforms state-of-the-art systems.

2 Related Work

There is a large body of work on WSD, covering supervised, unsupervised (word sense induction) and knowledge-based approaches (see McCarthy (2009) for an overview). While most supervised approaches treat the task as a classification task and use hand-labelled corpora as training data, most unsupervised systems automatically group word tokens into similar groups using clustering algorithms, and then assign labels to each sense cluster. Knowledge-based approaches exploit information contained in existing resources. They can be combined with supervised machine-learning models to assemble semi-supervised approaches.

Recently, a number of systems have been proposed that make use of topic models for sense disambiguation. Cai et al. (2007), for example, use LDA to capture global context. They compute topic models from a large unlabelled corpus and include them as features in a supervised system. Boyd-Graber and Blei (2007) propose an unsupervised approach that integrates McCarthy et al.'s (2004) method for finding predominant word senses into a topic modelling framework. In addition to generating a topic from the document's topic distribution and sampling a word from that topic, the enhanced model also generates a distributional neighbour for the chosen word and then assigns a sense based on the word, its neighbour and the topic. Boyd-Graber and Blei (2007) test their method on WSD and information retrieval tasks and find that it can lead to modest improvements over state-of-the-art results.

In another unsupervised system, Boyd-Graber et al. (2007) enhance the basic LDA algorithm by incorporating WordNet senses as an additional latent variable. Instead of generating words directly from a topic, each topic is associated with a random walk through the WordNet hierarchy which generates the observed word. Topics and synsets are then inferred together. While Boyd-Graber et al. (2007) show that this method can lead to improvements in accuracy, they also find that idiosyncracies in the hierarchical structure of WordNet can harm performance. This is a general problem for methods which use hierarchical lexicons to model semantic distance (Budanitsky and Hirst, 2006). In our approach, we circumvent this problem by exploiting paraphrase information for the target senses rather than relying on the structure of WordNet as a whole.

Topic models have also been applied to the related task of word sense induction. Brody and Lapata (2009) propose a method that integrates a number of different linguistic features into a single generative model.

Topic models have been previously considered for metaphor extraction and estimating the frequency of metaphors (Klebanov et al., 2009; Bethard et al., 2009). However, we have a different focus in this paper, which aims to distinguish literal and nonliteral usages of potential idiomatic expressions. A number of methods have been applied to this task. Katz and Giesbrecht (2006) devise a supervised method in which they compute the meaning vectors for the literal and nonliteral usages of a given expression in the trainning data. Birke and Sarkar (2006) use a clustering algorithm which compares test instances to two automatically constructed seed sets (one literal and one nonliteral), assigning the label of the closest set. An unsupervised method that computes cohesive links between the component words of the target expression and its context have been proposed (Sporleder and Li, 2009; Li and Sporleder, 2009). Their system predicts literal usages when strong links can be found.

3 The Sense Disambiguation Model

3.1 Topic Model

As pointed out by Hofmann (1999), the starting point of topic models is to decompose the conditional word-document probability distribution $p(w|d)$ into two different distributions: the word-topic distribution $p(w|z)$, and the topic-document distribution $p(z|d)$ (see Equation 1). This allows each semantic topic z to be represented as a multinominal distribution of words $p(w|z)$, and each document d to be represented as a multinominal distribution of semantic topics $p(z|d)$. The model introduces a conditional independence assumption that document d and word w are independent con-

ditioned on the hidden variable, topic z.

$$p(w|d) = \sum_z p(z|d)p(w|z) \quad (1)$$

LDA is a Bayesian version of this framework with Dirichlet hyper-parameters (Blei et al., 2003).

The inference of the two distributions given an observed corpus can be done through Gibbs Sampling (Geman and Geman, 1987; Griffiths and Steyvers, 2004). For each turn of the sampling, each word in each document is assigned a semantic topic based on the current word-topic distribution and topic-document distribution. The resulting topic assignments are then used to re-estimate a new word-topic distribution and topic-document distribution for the next turn. This process repeats until convergence. To avoid statistical coincidence, the final estimation of the distributions is made by the average of all the turns after convergence.

3.2 The Sense Disambiguation Model

Assigning the correct sense s to a target word w occurring in a context c involves finding the sense which maximizes the conditional probability of senses given a context:

$$s = \arg\max_{s_i} p(s_i|c) \quad (2)$$

In our model, we represent a sense (s_i) as a collection of 'paraphrases' that capture (some aspect of) the meaning of the sense. These paraphrases can be taken from an existing resource such as WordNet (Miller, 1995) or supplied by the user (see Section 4).

This conditional probability is decomposed by incorporating a hidden variable, topic z, introduced by the topic model. We propose three variations of the basic model, depending on how much background information is available, i.e., knowledge of the prior sense distribution available and type of sense paraphrases used. In Model I and Model II, the sense paraphrases are obtained from WordNet, and both the context and the sense paraphrases are treated as documents, $c = dc$ and $s = ds$.

WordNet is a fairly rich resource which provides detailed information about word senses (glosses, example sentences, synsets, semantic relations between senses, etc.). Sometimes such detailed information may not be available, for instance for languages for which such a resource does not exist or for expressions that are not very well covered in WordNet, such as idioms. For those situations, we propose another model, Model III, in which contexts are treated as documents while sense paraphrases are treated as sequences of independent words.[1]

Model I directly maximizes the conditional probability of the sense given the context, where the sense is modeled as a 'paraphrase document' ds and the context as a 'context document' dc. The conditional probability of sense given context $p(ds|dc)$ can be rewritten as a joint probability divided by a normalization factor:

$$p(ds|dc) = \frac{p(ds, dc)}{p(dc)} \quad (3)$$

This joint probability can be rewritten as a generative process by introducing a hidden variable z. We make the conditional independence assumption that, conditioned on the topic z, a paraphrase document ds is generated independently of the specific context document dc:

$$p(ds, dc) = \sum_z p(ds)p(z|ds)p(dc|z) \quad (4)$$

We apply the same process to the conditional probability $p(dc|z)$. It can be rewritten as:

$$p(dc|z) = \frac{p(dc)p(z|dc)}{p(z)} \quad (5)$$

Now, the disambiguation model $p(ds|dc)$ can be rewritten as a prior $p(ds)$ times a topic function $f(z)$:

$$p(ds|dc) = p(ds) \sum_z \frac{p(z|dc)p(z|ds)}{p(z)} \quad (6)$$

As $p(z)$ is a uniform distribution according to the uniform Dirichlet priors assumption, Equation 6 can be rewritten as:

$$p(ds|dc) \propto p(ds) \sum_z p(z|dc)p(z|ds) \quad (7)$$

Model I:

$$\arg\max_{ds_i} p(ds_i) \sum_z p(z|dc)p(z|ds_i) \quad (8)$$

Model I has the disadvantage that it requires information about the prior distribution of senses

[1] The idea is that these key words capture the meaning of the idioms.

$p(ds)$, which is not always available. We use sense frequency information from WordNet to estimate the prior sense distribution, although it must be kept in mind that, depending on the genre of the texts, it is possible that the distribution of senses in the testing corpus may diverge greatly from the WordNet-based estimation. If there is no means for estimating the prior sense distribution of an experimental corpus, generally a uniform distribution must be assumed. However, this assumption does not hold, as the true distribution of word senses is often highly skewed (McCarthy, 2009).

To overcome this problem, we propose Model II, which indirectly maximizes the sense-context probability by maximizing the cosine value of two document vectors that encode the document-topic frequencies from sampling, $v(z|dc)$ and $v(z|ds)$. The document vectors are represented by topics, with each dimension representing the number of times that the tokens in this document are assigned to a certain topic.

Model II:

$$\arg\max_{ds_i} \cos(v(z|dc), v(z|ds_i)) \tag{9}$$

If the prior distribution of senses is known, Model I is the best choice. However, Model II has to be chosen instead when this knowledge is not available. In our experiments, we test the performance of both models (see Section 5).

If the sense paraphrases are very short, it is difficult to reliably estimate $p(z|ds)$. In order to solve this problem, we treat the sense paraphrase ds as a 'query', a concept which is used in information retrieval. One model from information retrieval takes the conditional probability of the query given the document as a product of all the conditional probabilities of words in the query given the document. The assumption is that the query is generated by a collection of conditionally independent words (Song and Croft, 1999).

We make the same assumption here. However, instead of taking the product of all the conditional probabilities of words given the document, we take the maximum. There are two reasons for this: (i) taking the product may penalize longer paraphrases since the product of probabilities decreases as there are more words; (ii) we do not want to model the probability of generating specific paraphrases, but rather the probability of generating a sense, which might only be represented by one or two words in the paraphrases (e.g., the potentially idiomatic phrase 'rock the boat' can be paraphrased as 'break the norm' or 'cause trouble'. A similar topic distribution to that of the individual words 'norm' or 'trouble' would be strong supporting evidence of the corresponding idiomatic reading.). We propose **Model III**:

$$\arg\max_{qs_i} \max_{w_i \in qs} \sum_z p(w_i|z)p(z|dc) \tag{10}$$

where qs is a collection of words contained in the sense paraphrases.

3.3 Inference

One possible inference approach is to combine the context documents and sense paraphrases into a corpus and run Gibbs sampling on this corpus. The problem with this approach is that the test set and sense paraphrase set are relatively small, and topic models running on a small corpus are less likely to capture rich semantic topics. One simple explanation for this is that a small corpus usually has a relatively small vocabulary, which is less representative of topics, i.e., $p(w|z)$ cannot be estimated reliably.

In order to overcome this problem, we infer the word-topic distribution from a very large corpus (Wikipedia dump, see Section 4). All the following inference experiments on the test corpus are based on the assumption that the word-topic distribution $p(w|z)$ is the same as the one estimated from the Wikipedia dump. Inference of topic-document distributions for context and sense paraphrases is done by fixing the word-topic distribution as a constant.

4 Experimental Setup

We evaluate our models on three different tasks: coarse-grained WSD, fine-grained WSD and literal vs. nonliteral sense detection. In this section we discuss our experimental set-up. We start by describing the three datasets for evaluation and another dataset for probability estimation. We also discuss how we choose sense paraphrases and instance contexts.

Data We use three datasets for evaluation. The coarse-grained task is evaluated on the Semeval-2007 Task-07 benchmark dataset released by Navigli et al. (2009). The dataset consists of 5377 words of running text from five different articles: the first three were obtained from the WSJ corpus, the fourth was the Wikipedia entry for *computer programming*, and the fifth was an excerpt of

Amy Steedman's *Knights of the Art*, biographies of Italian painters. The proportion of the non news text, the last two articles, constitutes 51.87% of the whole testing set. It consists of 1108 nouns, 591 verbs, 362 adjectives, and 208 adverbs. The data were annotated with coarse-grained senses which were obtained by clustering senses from the WordNet 2.1 sense inventory based on the procedure proposed by Navigli (2006).

To determine whether our model is also suitable for fine-grained WSD, we test on the data provided by Pradhan et al. (2009) for the Semeval-2007 Task-17 (English fine-grained all-words task). This dataset is a subset of the set from Task-07. It comprises the three WSJ articles from Navigli et al. (2009). A total of 465 lemmas were selected as instances from about 3500 words of text. There are 10 instances marked as 'U' (undecided sense tag). Of the remaining 455 instances, 159 are nouns and 296 are verbs. The sense inventory is from WordNet 2.1.

Finally, we test our model on the related sense disambiguation task of distinguishing literal and nonliteral usages of potentially ambiguous expressions such as *break the ice*. For this, we use the dataset from Sporleder and Li (2009) as a test set. This dataset consists of 3964 instances of 17 potential English idioms which were manually annotated as *literal* or *nonliteral*.

A Wikipedia dump[2] is used to estimate the multinomial word-topic distribution. This dataset, which consists of 320,000 articles,[3] is significantly larger than SemCor, which is the dataset used by Boyd-Graber et al. (2007). All markup from the Wikipedia dump was stripped off using the same filter as the ESA implementation (Sorg and Cimiano, 2008), and stopwords were filtered out using the Snowball (Porter, October 2001) stopword list. In addition, words with a Wikipedia document frequency of 1 were filtered out. The lemmatized version of the corpus consists of 299,825 lexical units.

The test sets were POS-tagged and lemmatized using RASP (Briscoe and Carroll, 2006). The inference processes are run on the lemmatized version of the corpus. For the Semeval-2007 Task 17 English all-words, the organizers do not supply the part-of-speech and lemma information of the target instances. In order to avoid the wrong predictions caused by tagging or lemmatization errors, we manually corrected any bad tags and lemmas for the target instances.[4]

Sense Paraphrases For word sense disambiguation tasks, the paraphrases of the sense keys are represented by information from WordNet 2.1. (Miller, 1995). To obtain the paraphrases, we use the **word forms**, **glosses** and **example sentences** of the *synset itself* and a set of selected *reference synsets* (i.e., synsets linked to the target synset by specific semantic relations, see Table 1). We excluded the 'hypernym reference synsets', since information common to all of the child synsets may confuse the disambiguation process.

For the literal vs. nonliteral sense detection task, we selected the paraphrases of the nonliteral meaning from several online idiom dictionaries. For the literal senses, we used 2-3 manually selected words with which we tried to capture (aspects of) the literal meaning of the expression.[5] For instance, the literal 'paraphrases' that we chose for 'break the ice' were *ice, water* and *snow*. The paraphrases are shorter for the idiom task than for the WSD task, because the meaning descriptions from the idiom dictionaries are shorter than what we get from WordNet. In the latter case, each sense can be represented by its synset as well as its reference synsets.

Instance Context We experimented with different context sizes for the disambiguation task. The five different context settings that we used for the WSD tasks are: collocations (1w), $\pm$5-word window (5w), $\pm$10-word window (10w), current sentence, and whole text. Because the idiom corpus also includes explicitly marked paragraph boundaries, we included 'paragraph' as a sixth type of context size for the idiom sense detection task.

5 Experiments

As mentioned above, we test our proposed sense disambiguation framework on three tasks. We start by describing the sampling experiments for

[2]We use the English snapshot of 2009-07-13

[3]All articles of fewer than 100 words were discarded.

[4]This was done by comparing the predicted sense keys and the gold standard sense keys. We only checked instances for which the POS-tags in the predicted sense keys are not consistent with those in the gold standard. This was the case for around 20 instances.

[5]Note that we use the word 'paraphrase' in a fairly wide sense in this paper. Sometimes it is not possible to obtain exact paraphrases. This applies especially to the task of distinguishing literal from nonliteral senses of multi-word expressions. In this case we take as paraphrases some key words which capture salient aspects of the meaning.

POS	Paraphrase reference synsets
N	hyponyms, instance hyponyms, member holonyms, substance holonyms, part holonyms, member meronyms, part meronyms, substance meronyms, attributes, topic members, region members, usage members, topics, regions, usages
V	Troponyms, entailments, outcomes, phrases, verb groups, topics, regions, usages, sentence frames
A	similar, pertainym, attributes, related, topics, regions, usages
R	pertainyms, topics, regions, usages

Table 1: Selected reference synsets from WordNet that were used for different parts-of-speech to obtain word sense paraphrase. N(noun), V(verb), A(adj), R(adv).

estimating the word-topic distribution from the Wikipedia dump. We used the package provided by Wang et al. (2009) with the suggested Dirichlet hyper-parameters [6]. In order to avoid statistical instability, the final result is averaged over the last 50 iterations. We did four rounds of sampling with 1000, 500, 250, and 125 topics respectively. The final word-topic distribution is a normalized concatenate of the four distributions estimated in each round. In average, the sampling program run on the Wikipedia dump consumed 20G memory, and each round took about one week on a single AMD Dual-Core 1000MHZ processor.

5.1 Coarse-Grained WSD

In this section we first describe the landscape of similar systems against which we compare our models, then present the results of the comparison. The systems that participated in the SemEval-2007 coarse-grained WSD task (Task-07) can be divided into three categories, depending on whether training data is needed and whether other types of background knowledge are required: What we call Type I includes all the systems that need annotated training data. All the participating systems that have the mark *TR* fall into this category (see Navigli et al. (2009) for the evaluation for all the participating systems). Type II consists of systems that do not need training data but require prior knowledge of the sense distribution (estimated sense frequency). All the participating systems that have the mark *MFS* belong to this category. Systems that need neither training data nor prior sense distribution knowledge are categorized as Type III.

We make this distinction based on two principles: (i) the cost of building a system; (ii) the portability of the established resource. Type III is the cheapest system to build, while Type I and Type II both need extra resources. Type II has an advantage over Type I since the prior knowledge of the sense distribution can be estimated from annotated corpora (e.g.: SemCor, Senseval). In contrast, training data in Type I may be system specific (e.g.: different input format, different annotation guidelines). McCarthy (2009) also addresses the issue of performance and cost by comparing supervised word sense disambiguation systems with unsupervised ones.

We exclude the system provided by one of the organizers (UoR-SSI) from our categorization. The reason is that although this system is claimed to be unsupervised, and it performs better than all the participating systems (including the supervised systems) in the SemEval-2007 shared task, it still needs to incorporate a lot of prior knowledge, specifically information about co-occurrences between different word senses, which was obtained from a number of resources (SSI+LKB) including: (i) SemCor (manually annotated); (ii) LDC-DSO (partly manually annotated); (iii) collocation dictionaries which are then disambiguated semi-automatically. Even though the system is not "trained", it needs a lot of information which is largely dependent on manually annotated data, so it does not fit neatly into the categories Type II or Type III either.

Table 2 lists the best participating systems of each type in the SemEval-2007 task (Type I: NUS-PT (Chan et al., 2007); Type II: UPV-WSD (Buscaldi and Rosso, 2007); Type III: TKB-UO (Anaya-Sánchez et al., 2007)). Our Model I belongs to Type II, and our Model II belongs to Type III.

Table 2 compares the performance of our models with the Semeval-2007 participating systems. We only compare the F-score, since all the compared systems have an attempted rate[7] of 1.0,

[6] They were set as: $\alpha = \frac{50}{\#topics}$ and $\beta = 0.01$.

[7] Attempted rate is defined as the total number of disambiguated output instances divided by the total number of input

which makes both the precision and recall rates the same as the F-score. We focus on comparisons between our models and the best SemEval-2007 participating systems within the same type. Model I is compared with UPV-WSD, and Model II is compared with TKB-UO. In addition, we also compare our system with the most frequent sense baseline which was not outperformed by any of the systems of Type II and Type III in the SemEval-2007 task.

Comparison on Type III is marked with $'$, while comparison on Type II is marked with $*$. We find that Model II performs statistically significantly better than the best participating system of the same type TKB-UO ($p<<0.01$, χ^2 test). When encoded with the prior knowledge of sense distribution, Model I outperforms by 1.36% the best Type II system UPV-WSD, although the difference is not statistically significant. Furthermore, Model I also quantitatively outperforms the most frequent sense baseline BL_{mfs}, which, as mentioned above, was not beat by any participating systems that do not use training data.

We also find that our model works best for nouns. The unsupervised Type III model Model II achieves better results than the most frequent sense baseline on nouns, but not on other parts-of-speech. This is in line with results obtained by previous systems (Griffiths et al., 2005; Boyd-Graber and Blei, 2008; Cai et al., 2007). While the performance on verbs can be increased to outperform the most frequent sense baseline by including the prior sense probability, the performance on adjectives and adverbs remains below the most frequent sense baseline. We think that there are three reasons for this: first, adjectives and adverbs have fewer reference synsets for paraphrases compared with nouns and verbs (see Table 1); second, adjectives and adverbs tend to convey less key semantic content in the document, so they are more difficult to capture by the topic model; and third, adjectives and adverbs are a small portion of the test set, so their performances are statistically unstable. For example, if 'already' appears 10 times out of 20 adverb instances, a system may get bad result on adverbs only because of its failure to disambiguate the word 'already'.

Paraphrase analysis Table 2 also shows the effect of different ways of choosing sense paraphrases. MII+ref is the result of including the reference synsets, while MII-ref excludes the reference synsets. As can be seen from the table, including all reference synsets in sense paraphrases increases performance. Longer paraphrases contain more information, and they are statistically more stable for inference.

instances.

System	Noun	Verb	Adj	Adv	All
UoR-SSI	84.12	78.34	85.36	88.46	83.21
NUS-PT	82.31	78.51	85.64	89.42	82.50
UPV-WSD	79.33	72.76	84.53	81.52	**78.63***
TKB-UO	70.76	62.61	78.73	74.04	**70.21**$'$
MII–ref	78.16	70.39	79.56	81.25	76.64
MII+ref	80.05	70.73	82.04	82.21	**78.14**$'$
MI+ref	79.96	75.47	83.98	86.06	**79.99***
BL_{mfs}	77.44	75.30	84.25	87.50	**78.99***

Table 2: Model performance (F-score) on the coarse-grained dataset (context=sentence). Paraphrases with/without reference synsets (+ref/-ref).

Context	Ate.	Pre.	Rec.	F1
$\pm 1w$	91.67	75.05	68.80	71.79
$\pm 5w$	99.29	77.14	76.60	76.87
$\pm 10w$	100	77.92	77.92	77.92
text	100	76.86	76.86	76.86
sent.	100	**78.14**	**78.14**	**78.14**

Table 3: Model II performance on different context size. attempted rate (Ate.), precision (Pre.), recall (Rec.), F-score (F1).

We find that nouns get the greatest performance boost from including reference synsets, as they have the largest number of different types of synsets. We also find the 'similar' reference synset for adjectives to be very useful. Performance on adjectives increases by 2.75% when including this reference synset.

Context analysis In order to study how the context influences the performance, we experiment with Model II on different context sizes (see Table 3). We find *sentence context* is the best size for this disambiguation task. Using a smaller context not only reduces the precision, but also reduces the recall rate, which is caused by the all-zero topic assignment by the topic model for documents only containing words that are not in the vocabulary. As a result, the model is unable to disambiguate. The context based on the whole text (article) does not perform well either, possibly because using the full text folds in too much noisy information.

System	F-score
RACAI	52.7 ±4.5
BL_{mfs}	55.91±4.5
MI+ref	**56.99**±4.5

Table 4: Model performance (F-score) for the fine-grained word sense disambiguation task.

5.2 Fine-grained WSD

We saw in the previous section that our framework performs well on coarse-grained WSD. Fine-grained WSD, however, is a more difficult task. To determine whether our framework is also able to detect subtler sense distinctions, we tested Model I on the English all-words subtask of SemEval-2007 Task-17 (see Table 4).

We find that Model I performs better than both the best unsupervised system, RACAI (Ion and Tufiş, 2007) and the most frequent sense baseline (BL_{mfs}), although these differences are not statistically significant due to the small size of the available test data (465).

5.3 Idiom Sense Disambiguation

In the previous section, we provided the results of applying our framework to coarse- and fine-grained word sense disambiguation tasks. For both tasks, our models outperform the state-of-the-art systems of the same type either quantitatively or statistically significantly. In this section, we apply Model III to another sense disambiguation task, namely distinguishing literal and nonliteral senses of ambiguous expressions.

WordNet has a relatively low coverage for idiomatic expressions. In order to represent nonliteral senses, we replace the paraphrases obtained automatically from WordNet by words selected manually from online idiom dictionaries (for the nonliteral sense) and by linguistic introspection (for the literal sense). We then compare the topic distributions of literal and nonliteral senses.

As the paraphrases obtained from the idiom dictionary are very short, we treat the paraphrase as a sequence of independent words instead of as a document and apply Model III (see Section 3). Table 5 shows the results of our proposed model compared with state-of-the-art systems. We find that the system significantly outperforms the majority baseline ($p<<0.01$, χ^2 test) and the cohesion-graph based approach proposed by Sporleder and Li (2009) ($p<<0.01$, χ^2 test). The system also outperforms the bootstrapping system by Li and Sporleder (2009), although not statistically significantly. This shows how a limited amount of human knowledge (e.g., paraphrases) can be added to an unsupervised system for a strong boost in performance (Model III compared with the cohesion-graph and the bootstrapping approaches).

System	$Prec_l$	Rec_l	F_l	Acc.
$Base_{maj}$	-	-	-	78.25
co-graph	50.04	69.72	58.26	78.38
boot.	71.86	66.36	69.00	87.03
Model III	67.05	81.07	73.40	87.24

Table 5: Performance on the literal or nonliteral sense disambiguation task on idioms. literal precision ($Prec_l$), literal recall (Rec_l), literal F-score (F_l), accuracy(Acc.).

For obvious reasons, this approach is sensitive to the quality of the paraphrases. The paraphrases chosen to characterise (aspects of) the meaning of a sense should be non-ambiguous between the literal or idiomatic meaning. For instance, 'fire' is not a good choice for a paraphrase of the literal reading of 'play with fire', since this word can be interpreted literally as 'fire' or metaphorically as 'something dangerous'. The verb component word 'play' is a better literal paraphrase.

For the same reason, this approach works well for expressions where the literal and nonliteral readings are well separated (i.e., occur in different contexts), while the performance drops for expressions whose literal and idiomatic readings can appear in a similar context. We test the performance on individual idioms on the five most frequent idioms in our corpus[8] (see Table 6). We find that 'drop the ball' is a difficult case. The words 'fault', 'mistake', 'fail' or 'miss' can be used as the nonliteral paraphrases. However, it is also highly likely that these words are used to describe a scenario in a baseball game, in which 'drop the ball' is used literally. In contrast, the performance on 'rock the boat' is much better, since the nonliteral reading of the phrases 'break the norm' or 'cause trouble' are less likely to be linked with the literal reading 'boat'. This may also be because 'boat' is not often used metaphorically in the corpus.

As the topic distribution of nouns and verbs exhibit different properties, topic comparisons across parts-of-speech do not make sense. We

[8]We tested only on the most frequent idioms in order to avoid statistically unreliable observations.

Idiom	Acc.
drop the ball	75.86
play with fire	91.17
break the ice	87.43
rock the boat	95.82
set in stone	89.39

Table 6: Performance on individual idioms.

make the topic distributions comparable by making sure each type of paraphrase contains the same sets of parts-of-speech. For instance, we do not permit combinations of literal paraphrases which only consist of nouns and nonliteral paraphrases which only consist of verbs.

6 Conclusion

We propose three models for sense disambiguation on words and multi-word expressions. The basic idea of these models is to compare the topic distribution of a target instance with the candidate sense paraphrases and choose the most probable one. While Model I and Model III model the problem in a probabilistic way, Model II uses a vector space model by comparing the cosine values of two topic vectors. Model II and Model III are completely unsupervised, while Model I needs the prior sense distribution. Model I and Model II treat the sense paraphrases as documents, while Model III treats the sense paraphrases as a collection of independent words.

We test the proposed models on three tasks. We apply Model I and Model II to the WSD tasks due to the availability of more paraphrase information. Model III is applied to the idiom detection task since the paraphrases from the idiom dictionary are smaller. We find that all models outperform comparable state-of-the-art systems either quantitatively or statistically significantly.

By testing our framework on three different sense disambiguation tasks, we show that the framework can be used flexibly in different application tasks. The system also points out a promising way of solving the granularity problem of word sense disambiguation, as new application tasks which need different sense granularities can utilize this framework when new paraphrases of sense clusters are available. In addition, this system can also be used in a larger context such as figurative language identification (*literal* or *figurative*) and sentiment detection (*positive* or *negative*).

Acknowledgments

This work was funded by the DFG within the Cluster of Excellence "Multimodal Computing and Interaction".

References

H. Anaya-Sánchez, A. Pons-Porrata, R. Berlanga-Llavori. 2007. TKB-UO: using sense clustering for WSD. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, 322–325.

S. Bethard, V. T. Lai, J. H. Martin. 2009. Topic model analysis of metaphor frequency for psycholinguistic stimuli. In *CALC '09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, 9–16, Morristown, NJ, USA. Association for Computational Linguistics.

J. Birke, A. Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *Proceedings of EACL-06*.

D. M. Blei, A. Y. Ng, M. I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Reseach*, 3:993–1022.

J. Boyd-Graber, D. Blei. 2007. PUTOP: turning predominant senses into a topic model for word sense disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, 277–281.

J. Boyd-Graber, D. Blei. 2008. Syntactic topic models. *Computational Linguistics*.

J. Boyd-Graber, D. Blei, X. Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 1024–1033.

T. Briscoe, J. Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL on Main conference poster sessions*, 41–48.

S. Brody, M. Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, 103–111.

A. Budanitsky, G. Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

D. Buscaldi, P. Rosso. 2007. UPV-WSD: Combining different WSD methods by means of Fuzzy Borda Voting. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, 434–437.

J. Cai, W. S. Lee, Y. W. Teh. 2007. Improving word sense disambiguation using topic features. In *Proceedings of the 2007 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 1015–1023.

Y. S. Chan, H. T. Ng, Z. Zhong. 2007. NUS-PT: exploiting parallel texts for word sense disambiguation in the English all-words tasks. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, 253–256.

S. Geman, D. Geman. 1987. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision: issues, problems, principles, and paradigms*, 564–584. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

T. L. Griffiths, M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

T. L. Griffiths, M. Steyvers, D. M. Blei, J. B. Tenenbaum. 2005. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, 537–544. MIT Press.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 50–57.

R. Ion, D. Tufiş. 2007. Racai: meaning affinity models. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, 282–287, Morristown, NJ, USA. Association for Computational Linguistics.

G. Katz, E. Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the ACL/COLING-06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*.

B. B. Klebanov, E. Beigman, D. Diermeier. 2009. Discourse topics and metaphors. In *CALC '09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

L. Li, C. Sporleder. 2009. Contextual idiom detection without labelled data. In *Proceedings of EMNLP-09*.

D. McCarthy, R. Koeling, J. Weeds, J. Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, 279–286.

D. McCarthy. 2009. Word sense disambiguation: An overview. *Language and Linguistics Compass*, 3(2):537–558.

G. A. Miller. 1995. WordNet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

R. Navigli, K. C. Litkowski, O. Hargraves. 2009. SemEval-2007 Task 07: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval-2007)*.

R. Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 44th Annual Meeting of the Association for Computational Liguistics joint with the 21st International Conference on Computational Liguistics (COLING-ACL 2006)*.

M. Porter. October 2001. Snowball: A language for stemming algorithms. `http://snowball.tartarus.org/texts/introduction.html`.

S. S. Pradhan, E. Loper, D. Dligach, M. Palmer. 2009. SemEval-2007 Task 07: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval-2007)*.

F. Song, W. B. Croft. 1999. A general language model for information retrieval (poster abstract). In *Research and Development in Information Retrieval*, 279–280.

P. Sorg, P. Cimiano. 2008. Cross-lingual information retrieval with explicit semantic analysis. In *In Working Notes for the CLEF 2008 Workshop*.

C. Sporleder, L. Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of EACL-09*.

Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, E. Y. Chang. 2009. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Proc. of 5th International Conference on Algorithmic Aspects in Information and Management*. Software available at `http://code.google.com/p/plda`.

PCFGs, Topic Models, Adaptor Grammars and Learning Topical Collocations and the Structure of Proper Names

Mark Johnson
Department of Computing
Macquarie University
mjohnson@science.mq.edu.au

Abstract

This paper establishes a connection between two apparently very different kinds of probabilistic models. Latent Dirichlet Allocation (LDA) models are used as "topic models" to produce a low-dimensional representation of documents, while Probabilistic Context-Free Grammars (PCFGs) define distributions over trees. The paper begins by showing that LDA topic models can be viewed as a special kind of PCFG, so Bayesian inference for PCFGs can be used to infer Topic Models as well. Adaptor Grammars (AGs) are a hierarchical, non-parameteric Bayesian extension of PCFGs. Exploiting the close relationship between LDA and PCFGs just described, we propose two novel probabilistic models that combine insights from LDA and AG models. The first replaces the unigram component of LDA topic models with multi-word sequences or collocations generated by an AG. The second extension builds on the first one to learn aspects of the internal structure of proper names.

1 Introduction

Over the last few years there has been considerable interest in Bayesian inference for complex hierarchical models both in machine learning and in computational linguistics. This paper establishes a theoretical connection between two very different kinds of probabilistic models: Probabilistic Context-Free Grammars (PCFGs) and a class of models known as Latent Dirichlet Allocation (Blei et al., 2003; Griffiths and Steyvers, 2004) models that have been used for a variety of tasks in machine learning. Specifically, we show that an LDA model can be expressed as a certain kind of PCFG, so Bayesian inference for PCFGs can be used to learn LDA topic models as well. The importance of this observation is primarily theoretical, as current Bayesian inference algorithms for PCFGs are less efficient than those for LDA inference. However, once this link is established it suggests a variety of extensions to the LDA topic models, two of which we explore in this paper. The first involves extending the LDA topic model so that it generates collocations (sequences of words) rather than individual words. The second applies this idea to the problem of automatically learning internal structure of proper names (NPs), which is useful for definite NP coreference models and other applications.

The rest of this paper is structured as follows. The next section reviews Latent Dirichlet Allocation (LDA) topic models, and the following section reviews Probabilistic Context-Free Grammars (PCFGs). Section 4 shows how an LDA topic model can be expressed as a PCFG, which provides the fundamental connection between LDA and PCFGs that we exploit in the rest of the paper, and shows how it can be used to define a "sticky topic" version of LDA. The following section reviews Adaptor Grammars (AGs), a non-parametric extension of PCFGs introduced by Johnson et al. (2007b). Section 6 exploits the connection between LDA and PCFGs to propose an AG-based topic model that extends LDA by defining distributions over collocations rather than individual words, and section 7 applies this extension to the problem of finding the structure of proper names.

2 Latent Dirichlet Allocation Models

Latent Dirichlet Allocation (LDA) was introduced as an explicit probabilistic counterpart to Latent Semantic Indexing (LSI) (Blei et al., 2003). Like LSI, LDA is intended to produce a low-dimensional characterisation or summary of a doc-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1148–1157,
Uppsala, Sweden, 11-16 July 2010.

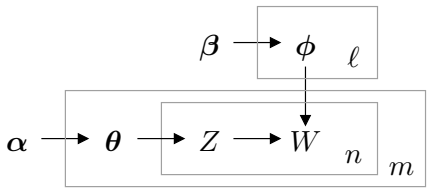


Figure 1: A graphical model "plate" representation of an LDA topic model. Here ℓ is the number of topics, m is the number of documents and n is the number of words per document.

ument in a collection of documents for information retrieval purposes. Both LSI and LDA do this by mapping documents to points in a relatively low-dimensional real-valued vector space; distance in this space is intended to correspond to document similarity.

An LDA model is an explicit generative probabilistic model of a collection of documents. We describe the "smoothed" LDA model here (see page 1006 of Blei et al. (2003)) as it corresponds precisely to the Bayesian PCFGs described in section 4. It generates a collection of documents by first generating multinomials $\boldsymbol{\phi}_i$ over the vocabulary V for each topic $i \in 1, \ldots, \ell$, where ℓ is the number of topics and $\phi_{i,w}$ is the probability of generating word w in topic i. Then it generates each document $D_j, j = 1, \ldots, m$ in turn by first generating a multinomial $\boldsymbol{\theta}_j$ over topics, where $\theta_{j,i}$ is the probability of topic i appearing in document j. ($\boldsymbol{\theta}_j$ serves as the low-dimensional representation of document D_j). Finally it generates each of the n words of document D_j by first selecting a topic z for the word according to $\boldsymbol{\theta}_j$, and then drawing a word from $\boldsymbol{\phi}_z$. Dirichlet priors with parameters $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ respectively are placed on the $\boldsymbol{\phi}_i$ and the $\boldsymbol{\theta}_j$ in order to avoid the zeros that can arise from maximum likelihood estimation (i.e., sparse data problems).

The LDA generative model can be compactly expressed as follows, where "$\sim$" should be read as "is distributed according to".

$$\begin{array}{rcll} \boldsymbol{\phi}_i & \sim & \mathrm{Dir}(\boldsymbol{\beta}) & i = 1, \ldots, \ell \\ \boldsymbol{\theta}_j & \sim & \mathrm{Dir}(\boldsymbol{\alpha}) & j = 1, \ldots, m \\ z_{j,k} & \sim & \boldsymbol{\theta}_j & j = 1, \ldots, m; k = 1, \ldots, n \\ w_{j,k} & \sim & \boldsymbol{\phi}_{z_{j,k}} & j = 1, \ldots, m; k = 1, \ldots, n \end{array}$$

In inference, the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of the Dirichlet priors are either fixed (i.e., chosen by the model designer), or else themselves inferred, e.g., by Bayesian inference. (The adaptor grammar software we used in the experiments described below automatically does this kind of hyper-parameter inference).

The inference task is to find the topic probability vector $\boldsymbol{\theta}_j$ of each document D_j given the words $w_{j,k}$ of the documents; in general this also requires inferring the topic to word distributions $\boldsymbol{\phi}$ and the topic assigned to each word $z_{j,k}$. Blei et al. (2003) describe a Variational Bayes inference algorithm for LDA models based on a mean-field approximation, while Griffiths and Steyvers (2004) describe an Markov Chain Monte Carlo inference algorithm based on Gibbs sampling; both are quite effective in practice.

3 Probabilistic Context-Free Grammars

Context-Free Grammars are a simple model of hierarchical structure often used to describe natural language syntax. A *Context-Free Grammar* (CFG) is a quadruple (N, W, R, S) where N and W are disjoint finite sets of *nonterminal* and *terminal* symbols respectively, R is a finite set of productions or *rules* of the form $A \rightarrow \beta$ where $A \in N$ and $\beta \in (N \cup W)^\star$, and $S \in N$ is the *start symbol*.

In what follows, it will be useful to interpret a CFG as generating sets of finite, labelled, ordered trees $\mathcal{T}_A$ for each $X \in N \cup W$. Informally, $\mathcal{T}_X$ consists of all trees t rooted in X where for each *local tree* (B, β) in t (i.e., where B is a parent's label and β is the sequence of labels of its immediate children) there is a rule $B \rightarrow \beta \in R$.

Formally, the sets $\mathcal{T}_X$ are the smallest sets of trees that satisfy the following equations.

If $X \in W$ (i.e., if X is a terminal) then $\mathcal{T}_X = \{X\}$, i.e., $\mathcal{T}_X$ consists of a single tree, which in turn only consists of a single node labelled X.

If $X \in N$ (i.e., if X is a nonterminal) then

$$\mathcal{T}_X = \bigcup_{X \rightarrow B_1 \ldots B_n \in R_X} \textsc{Tree}_X(\mathcal{T}_{B_1}, \ldots, \mathcal{T}_{B_n})$$

where $R_A = \{A \rightarrow \beta : A \rightarrow \beta \in R\}$ for each $A \in N$, and

$$\textsc{Tree}_X(\mathcal{T}_{B_1}, \ldots, \mathcal{T}_{B_n}) = \left\{ \overbrace{t_1 \ldots t_n}^{X} : \begin{array}{l} t_i \in \mathcal{T}_{B_i}, \\ i = 1, \ldots, n \end{array} \right\}$$

That is, $\textsc{Tree}_X(\mathcal{T}_{B_1}, \ldots, \mathcal{T}_{B_n})$ consists of the set of trees with whose root node is labelled X and whose ith child is a member of $\mathcal{T}_{B_i}$.

The set of trees generated by the CFG is $\mathcal{T}_S$, where S is the start symbol, and the set of strings generated by the CFG is the set of yields (i.e., terminal strings) of the trees in $\mathcal{T}_S$.

A *Probabilistic Context-Free Grammar* (PCFG) is a pair consisting of a CFG and set of multinomial probability vectors $\boldsymbol{\theta}_X$ indexed by nonterminals $X \in N$, where $\boldsymbol{\theta}_X$ is a distribution over the rules R_X (i.e., the rules expanding X). Informally, $\theta_{X\to\beta}$ is the probability of X expanding to β using the rule $X \to \beta \in R_X$. More formally, a PCFG associates each $X \in N \cup W$ with a distribution G_X over the trees $\mathcal{T}_X$ as follows.

If $X \in W$ (i.e., if X is a terminal) then G_X is the distribution that puts probability 1 on the single-node tree labelled X.

If $X \in N$ (i.e., if X is a nonterminal) then:

$$G_X = \sum_{X\to B_1 \ldots B_n \in R_X} \theta_{X\to B_1\ldots B_n} \mathrm{TD}_X(G_{B_1}, \ldots, G_{B_n}) \quad (1)$$

where:

$$\mathrm{TD}_A(G_1, \ldots, G_n)\left(\begin{array}{c} X \\ \diagup\ \diagdown \\ t_1\ \ldots\ t_n \end{array}\right) = \prod_{i=1}^{n} G_i(t_i).$$

That is, $\mathrm{TD}_A(G_1, \ldots, G_n)$ is a distribution over $\mathcal{T}_A$ where each subtree t_i is generated independently from G_i. These equations have solutions (i.e., the PCFG is said to be "consistent") when the rule probabilities $\boldsymbol{\theta}_A$ obey certain conditions; see e.g., Wetherell (1980) for details.

The PCFG generates the distribution over trees G_S, where S is the start symbol. The distribution over the strings it generates is obtained by marginalising over the trees.

In a Bayesian PCFG one puts Dirichlet priors $\mathrm{Dir}(\boldsymbol{\alpha}_X)$ on each of the multinomial rule probability vectors $\boldsymbol{\theta}_X$ for each nonterminal $X \in N$. This means that there is one Dirichlet parameter $\alpha_{X\to\beta}$ for each rule $X \to \beta \in R$ in the CFG.

In the "unsupervised" inference problem for a PCFG one is given a CFG, parameters $\boldsymbol{\alpha}_X$ for the Dirichlet priors over the rule probabilities, and a corpus of strings. The task is to infer the corresponding posterior distribution over rule probabilities $\boldsymbol{\theta}_X$. Recently Bayesian inference algorithms for PCFGs have been described. Kurihara and Sato (2006) describe a Variational Bayes algorithm for inferring PCFGs using a mean-field approximation, while Johnson et al. (2007a) describe a Markov Chain Monte Carlo algorithm based on Gibbs sampling.

4 LDA topic models as PCFGs

This section explains how to construct a PCFG that generates the same distribution over a collection of documents as an LDA model, and where Bayesian inference for the PCFG's rule probabilities yields the corresponding distributions as Bayesian inference of the corresponding LDA models. (There are several different ways of encoding LDA models as PCFGs; the one presented here is not the most succinct — it is possible to collapse the Doc and Doc' nonterminals — but it has the advantage that the LDA distributions map straight-forwardly onto PCFG nonterminals).

The terminals W of the CFG consist of the vocabulary V of the LDA model plus a set of special "document identifier" terminals "$\_j$" for each document $j \in 1, \ldots, m$, where m is the number of documents. In the PCFG encoding strings from document j are prefixed with "$\_j$"; this indicates to the grammar which document the string comes from. The nonterminals consist of the start symbol $\mathrm{Sentence}$, Doc_j and Doc'_j for each $j \in 1, \ldots, m$, and Topic_i for each $i \in 1, \ldots, \ell$, where ℓ is the number of topics in the LDA model.

The rules of the CFG are all instances of the following schemata:

$\mathrm{Sentence} \to \mathrm{Doc}'_j$	$j \in 1, \ldots, m$
$\mathrm{Doc}'_j \to \_j$	$j \in 1, \ldots, m$
$\mathrm{Doc}'_j \to \mathrm{Doc}'_j\ \mathrm{Doc}_j$	$j \in 1, \ldots, m$
$\mathrm{Doc}_j \to \mathrm{Topic}_i$	$i \in 1, \ldots, \ell; j \in 1, \ldots, m$
$\mathrm{Topic}_i \to w$	$i \in 1, \ldots, \ell; w \in V$

Figure 2 depicts a tree generated by such a CFG. The relationship between the LDA model and the PCFG can be understood by studying the trees generated by the CFG. In these trees the left-branching spine of nodes labelled Doc'_j propagate the document identifier throughout the whole tree. The nodes labelled Topic_i indicate the topics assigned to particular words, and the local trees expanding Doc_j to Topic_i (one per word in the document) indicate the distribution of topics in the document.

The corresponding Bayesian PCFG associates probabilities with each of the rules in the CFG. The probabilities $\boldsymbol{\theta}_{\mathrm{Topic}_i}$ associated with the rules expanding the Topic_i nonterminals indicate how words are distributed across topics; the $\boldsymbol{\theta}_{\mathrm{Topic}_i}$ probabilities correspond exactly to to the $\boldsymbol{\phi}_i$ probabilities in the LDA model. The probabilities

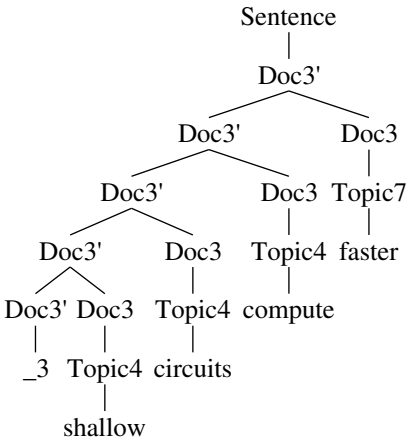


Figure 2: A tree generated by the CFG encoding an LDA topic model. The prefix "_3" indicates that this string belongs to document 3. The tree also indicates the assignment of words to topics.

$\boldsymbol{\theta}_{\mathrm{Doc}_j}$ associated with rules expanding Doc_j specify the distribution of topics in document j; they correspond exactly to the probabilities $\boldsymbol{\theta}_j$ of the LDA model. (The PCFG also specifies several other distributions that are suppressed in the LDA model. For example $\boldsymbol{\theta}_{\mathrm{Sentence}}$ specifies the distribution of documents in the corpus. However, it is easy to see that these distributions do not influence the topic distributions; indeed, the expansions of the $\mathrm{Sentence}$ nonterminal are completely determined by the document distribution in the corpus, and are not affected by $\boldsymbol{\theta}_{\mathrm{Sentence}}$).

A Bayesian PCFG places Dirichlet priors $\mathrm{Dir}(\boldsymbol{\alpha}_A)$ on the corresponding rule probabilities $\boldsymbol{\theta}_A$ for each $A \in N$. In the PCFG encoding an LDA model, the $\boldsymbol{\alpha}_{\mathrm{Topic}_i}$ parameters correspond exactly to the $\boldsymbol{\beta}$ parameters of the LDA model, and the $\boldsymbol{\alpha}_{\mathrm{Doc}_j}$ parameters correspond to the $\boldsymbol{\alpha}$ parameters of the LDA model.

As suggested above, each document D_j in the LDA model is mapped to a string in the corpus used to train the corresponding PCFG by prefixing it with a document identifier "$\_j$". Given this training data, the posterior distribution over rule probabilities $\theta_{\mathrm{Doc}_j \rightarrow \mathrm{Topic}_i}$ is the same as the posterior distribution over topics given documents $\theta_{j,i}$ in the original LDA model.

As we will see below, this connection between PCFGs and LDA topic models suggests a number of interesting variants of both PCFGs and topic models. Note that we are *not* suggesting that Bayesian inference for PCFGs is necessarily a good way of estimating LDA topic models. Current Bayesian PCFG inference algorithms require time proportional to the cube of the length of the longest string in the training corpus, and since these strings correspond to entire documents in our embedding, blindly applying a Bayesian PCFG inference algorithm is likely to be impractical.

A little reflection shows that the embedding still holds if the strings in the PCFG corpus correspond to sentences or even smaller units of the original document collection, so a single document would be mapped to multiple strings in the PCFG inference task. In this way the cubic time complexity of PCFG inference can be mitigated. Also, the trees generated by these CFGs have a very specialized left-branching structure, and it is straightforward to modify the general-purpose CFG inference procedures to avoid the cubic time complexity for such grammars: thus it may be practical to estimate topic models via grammatical inference.

However, we believe that the primary value of the embedding of LDA topic models into Bayesian PCFGs is theoretical: it suggests a number of novel extensions of both topic models and grammars that may be worth exploring. Our claim here is not that these models are the best algorithms for performing these tasks, but that the relationship we described between LDA models and PCFGs suggests a variety of interesting novel models.

We end this section with a simple example of such a modification to LDA. Inspired by the standard embedding of HMMs into PCFGs, we propose a "sticky topic" variant of LDA in which adjacent words are more likely to be assigned the same topic. Such an LDA extension is easy to describe as a PCFG (see Fox et al. (2008) for a similar model presented as an extended HMM). The nonterminals $\mathrm{Sentence}$ and Topic_i for $i = 1, \ldots, \ell$ have the same interpretation as before, but we introduce new nonterminals $\mathrm{Doc}_{j,i}$ that indicate we have just generated a nonterminal in document j belonging to topic i. Given a collection of m documents and ℓ topics, the rule schemata are as follows:

$\mathrm{Sentence} \rightarrow \mathrm{Doc}_{j,i}$	$i \in 1, \ldots, \ell;$ $j \in 1, \ldots, m$
$\mathrm{Doc}_{j,1} \rightarrow \_j$	$j \in 1, \ldots, m$
$\mathrm{Doc}_{j,i} \rightarrow \mathrm{Doc}_{j,i'}\ \mathrm{Topic}_i$	$i, i' \in 1, \ldots, \ell;$ $j \in 1, \ldots, m$
$\mathrm{Topic}_i \rightarrow w$	$i \in 1, \ldots, \ell; w \in V$

A sample parse generated by a "sticky topic"

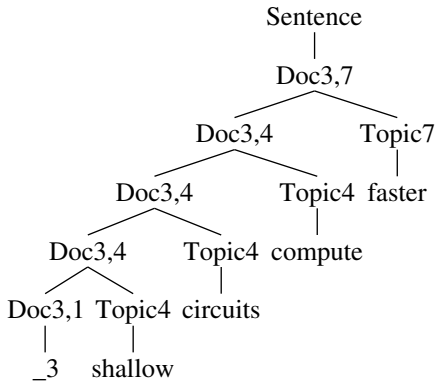


Figure 3: A tree generated by the "sticky topic" CFG. Here a nonterminal $\mathrm{Doc}3,7$ indicates we have just generated a word in document 3 belonging to topic 7.

CFG is shown in Figure 3. The probabilities of the rules $\mathrm{Doc}_{j,i} \rightarrow \mathrm{Doc}_{j,i'}\ \mathrm{Topic}_i$ in this PCFG encode the probability of shifting from topic i to topic i' (this PCFG can be viewed as generating the string from right to left).

We can use non-uniform sparse Dirichlet priors on the probabilities of these rules to encourage "topic stickiness". Specifically, by setting the Dirichlet parameters for the "topic shift" rules $\mathrm{Doc}_{j,i'} \rightarrow \mathrm{Doc}_{j,i}\ \mathrm{Topic}_i$ where $i' \neq i$ much lower than the parameters for the "topic preservation" rules $\mathrm{Doc}_{j,i} \rightarrow \mathrm{Doc}_{j,i}\ \mathrm{Topic}_i$, Bayesian inference will be biased to find distributions in which adjacent words will tend to have the same topic.

5 Adaptor Grammars

Non-parametric Bayesian inference, where the inference task involves learning not just the values of a finite vector of parameters but which parameters are relevant, has been the focus of intense research in machine learning recently. In the topic-modelling community this has lead to work on Dirichlet Processes and Chinese Restaurant Processes, which can be used to estimate the number of topics as well as their distribution across documents (Teh et al., 2006).

There are two obvious non-parametric extensions to PCFGs. In the first we regard the set of nonterminals N as potentially unbounded, and try to learn the set of nonterminals required to describe the training corpus. This approach goes under the name of the "infinite HMM" or "infinite PCFG" (Beal et al., 2002; Liang et al., 2007; Liang et al., 2009). Informally, we are given a set of "basic categories", say NP, VP, etc., and a set of rules that use these basic categories, say $\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP}$. The inference task is to learn a set of refined categories and rules (e.g., $\mathrm{S}_7 \rightarrow \mathrm{NP}_2\ \mathrm{VP}_5$) as well as their probabilities; this approach can therefore be viewed as a Bayesian version of the "split-merge" approach to grammar induction (Petrov and Klein, 2007).

In the second approach, which we adopt here, we regard the set of rules R as potentially unbounded, and try to learn the rules required to describe a training corpus as well as their probabilities. Adaptor grammars are an example of this approach (Johnson et al., 2007b), where entire subtrees generated by a "base grammar" can be viewed as distinct rules (in that we learn a separate probability for each subtree). The inference task is non-parametric if there are an unbounded number of such subtrees.

We review the adaptor grammar generative process below; for an informal introduction see Johnson (2008) and for details of the adaptor grammar inference procedure see Johnson and Goldwater (2009).

An adaptor grammar $(N, W, R, S, \theta, A, C)$ consists of a PCFG (N, W, R, S, θ) in which a subset $A \subseteq N$ of the nonterminals are *adapted*, and where each adapted nonterminal $X \in A$ has an associated adaptor C_X. An *adaptor* C_X for X is a function that maps a distribution over trees $\mathcal{T}_X$ to a distribution over distributions over $\mathcal{T}_X$ (we give examples of adaptors below).

Just as for a PCFG, an adaptor grammar defines distributions G_X over trees $\mathcal{T}_X$ for each $X \in N \cup W$. If $X \in W$ or $X \notin A$ then G_X is defined just as for a PCFG above, i.e., using (1). However, if $X \in A$ then G_X is defined in terms of an additional distribution H_X as follows:

$$\begin{aligned} G_X &\sim C_X(H_X) \\ H_X &= \sum_{X \rightarrow Y_1 \ldots Y_m \in R_X} \theta_{X \rightarrow Y_1 \ldots Y_m} \mathrm{TD}_X(G_{Y_1}, \ldots, G_{Y_m}) \end{aligned}$$

That is, the distribution G_X associated with an adapted nonterminal $X \in A$ is a sample from adapting (i.e., applying C_X to) its "ordinary" PCFG distribution H_X. In general adaptors are chosen for the specific properties they have. For example, with the adaptors used here G_X typically concentrates mass on a smaller subset of the trees $\mathcal{T}_X$ than H_X does.

Just as with the PCFG, an adaptor grammar generates the distribution over trees G_S, where $S \in N$

is the start symbol. However, while G_S in a PCFG is a fixed distribution (given the rule probabilities θ), in an adaptor grammar the distribution G_S is itself a random variable (because each G_X for $X \in A$ is random), i.e., an adaptor grammar generates a distribution over distributions over trees $\mathcal{T}_S$. However, the posterior joint distribution $\Pr(\boldsymbol{t})$ of a sequence $\boldsymbol{t} = (t_1, \ldots, t_n)$ of trees in $\mathcal{T}_S$ is well-defined:

$$\Pr(\boldsymbol{t}) \;=\; \int G_S(t_1) \ldots G_S(t_n)\, d\boldsymbol{G}$$

where the integral is over all of the random distributions $G_X, X \in A$. The adaptors we use in this paper are Dirichlet Processes or two-parameter Poisson-Dirichlet Processes, for which it is possible to compute this integral. One way to do this uses the predictive distributions:

$$\Pr(t_{n+1} \mid \boldsymbol{t}, H_X) \;\propto\; \int G_X(t_1) \ldots G_X(t_{n+1}) C_X(G_X \mid H_X)\, dG_X$$

where $\boldsymbol{t} = (t_1, \ldots, t_n)$ and each $t_i \in \mathcal{T}_X$. The predictive distribution for the Dirichlet Process is the (labeled) *Chinese Restaurant Process* (CRP), and the predictive distribution for the two-parameter Poisson-Dirichlet process is the (labeled) *Pitman-Yor Process* (PYP).

In the context of adaptor grammars, the CRP is:

$$\mathrm{CRP}(t \mid \boldsymbol{t}, \alpha_X, H_X) \;\propto\; n_t(\boldsymbol{t}) + \alpha_X H_X(t)$$

where $n_t(\boldsymbol{t})$ is the number of times t appears in $\boldsymbol{t}$ and $\alpha_X > 0$ is a user-settable "concentration parameter". In order to generate the next tree t_{n+1} a CRP either reuses a tree t with probability proportional to number of times t has been previously generated, or else it "backs off" to the "base distribution" H_X and generates a fresh tree t with probability proportional to $\alpha_X H_X(t)$.

The PYP is a generalization of the CRP:

$$\mathrm{PYP}(t \mid \boldsymbol{t}, a_X, b_X, H_X) \;\propto\; \max(0, n_t(\boldsymbol{t}) - m_t\, a_X) + (m a_X + b_X) H_X(t)$$

Here $a_X \in [0, 1]$ and $b_X > 0$ are user-settable parameters, and m_t is the number of times the PYP has generated t in $\boldsymbol{t}$ from the base distribution H_X, and $m = \sum_{t \in \mathcal{T}_X} m_t$ is the number of times any tree has been generated from H_X. (In the Chinese Restaurant metaphor, m_t is the number of tables labeled with t, and m is the number of occupied tables). If $a_X = 0$ then the PYP is equivalent to a CRP with $\alpha_X = b_X$, while if $a_X = 1$ then the PYP generates samples from H_X.

Informally, the CRP has a strong preference to regenerate trees that have been generated frequently before, leading to a "rich-get-richer" dynamics. The PYP can mitigate this somewhat by reducing the effective count of previously generated trees and redistributing that probability mass to new trees generated from H_X. As Goldwater et al. (2006) explain, Bayesian inference for H_X given samples from G_X is effectively performed from types if $a_X = 0$ and from tokens if $a_X = 1$, so varying a_X smoothly interpolates between type-based and token-based inference.

Adaptor grammars have previously been used primarily to study grammatical inference in the context of language acquisition. The *word segmentation task* involves segmenting a corpus of unsegmented phonemic utterance representations into words (Elman, 1990; Bernstein-Ratner, 1987). For example, the phoneme string corresponding to "you want to see the book" (with its correct segmentation indicated) is as follows:

y ▵ u ▴ w ▵ a ▵ n ▵ t ▴ t ▵ u ▴ s ▵ i ▴ D ▵ 6 ▴ b ▵ U ▵ k

We can represent any possible segmentation of any possible sentence as a tree generated by the following *unigram adaptor grammar*.

Sentence → <u>Word</u>
Sentence → <u>Word</u> Sentence
<u>Word</u> → Phonemes
Phonemes → Phoneme
Phonemes → Phoneme Phonemes

The trees generated by this adaptor grammar are the same as the trees generated by the CFG rules. For example, the following skeletal parse in which all but the Word nonterminals are suppressed (the others are deterministically inferrable) shows the parse that corresponds to the correct segmentation of the string above.

(Word y u) (Word w a n t) (Word t u)
(Word s i) (Word d 6) (Word b u k)

Because the Word nonterminal is *adapted* (indicated here by underlining) the adaptor grammar learns the probability of the entire Word subtrees (e.g., the probability that *b u k* is a Word); see Johnson (2008) for further details.

6 Topic models with collocations

Here we combine ideas from the unigram word segmentation adaptor grammar above and the PCFG encoding of LDA topic models to present a novel topic model that learns topical collocations. (For a non-grammar-based approach to this problem see Wang et al. (2007)). Specifically, we take the PCFG encoding of the LDA topic model described above, but modify it so that the Topic_i nodes generate sequences of words rather than single words. Then we adapt each of the Topic_i nonterminals, which means that we learn the probability of each of the sequences of words it can expand to.

$$
\begin{array}{ll}
\text{Sentence} \rightarrow \text{Doc}_j & j \in 1, \ldots, m \\
\text{Doc}_j \rightarrow \_j & j \in 1, \ldots, m \\
\text{Doc}_j \rightarrow \text{Doc}_j\ \text{Topic}_i & i \in 1, \ldots, \ell; \\
 & j \in 1, \ldots, m \\
\underline{\text{Topic}}_i \rightarrow \text{Words} & i \in 1, \ldots, \ell \\
\text{Words} \rightarrow \text{Word} & \\
\text{Words} \rightarrow \text{Words Word} & \\
\text{Word} \rightarrow w & w \in V
\end{array}
$$

In order to demonstrate that this model works, we implemented this using the publically-available adaptor grammar inference software,[1] and ran it on the NIPS corpus (composed of published NIPS abstracts), which has previously been used for studying collocation-based topic models (Griffiths et al., 2007). Because there is no generally accepted evaluation for collocation-finding, we merely present some of the sample analyses found by our adaptor grammar. We ran our adaptor grammar with $\ell = 20$ topics (i.e., 20 distinct Topic_i nonterminals). Adaptor grammar inference on this corpus is actually relatively efficient because the corpus provided by Griffiths et al. (2007) is already segmented by punctuation, so the terminal strings are generally rather short. Rather than set the Dirichlet parameters by hand, we placed vague priors on them and estimated them as described in Johnson and Goldwater (2009).

The following are some examples of collocations found by our adaptor grammar:

$\text{Topic}_0 \rightarrow$ cost function
$\text{Topic}_0 \rightarrow$ fixed point
$\text{Topic}_0 \rightarrow$ gradient descent
$\text{Topic}_0 \rightarrow$ learning rates
$\text{Topic}_1 \rightarrow$ associative memory
$\text{Topic}_1 \rightarrow$ hamming distance
$\text{Topic}_1 \rightarrow$ randomly chosen
$\text{Topic}_1 \rightarrow$ standard deviation
$\text{Topic}_3 \rightarrow$ action potentials
$\text{Topic}_3 \rightarrow$ membrane potential
$\text{Topic}_3 \rightarrow$ primary visual cortex
$\text{Topic}_3 \rightarrow$ visual system
$\text{Topic}_{10} \rightarrow$ nervous system
$\text{Topic}_{10} \rightarrow$ action potential
$\text{Topic}_{10} \rightarrow$ ocular dominance
$\text{Topic}_{10} \rightarrow$ visual field

The following are skeletal sample parses, where we have elided all but the adapted nonterminals (i.e., all we show are the Topic nonterminals, since the other structure can be inferred deterministically). Note that because Griffiths et al. (2007) segmented the NIPS abstracts at punctuation symbols, the training corpus contains more than one string from each abstract.

_3 (Topic_5 polynomial size)
(Topic_{15} threshold circuits)

_4 (Topic_{11} studied)
(Topic_{19} pattern recognition algorithms)

_4 (Topic_2 feedforward neural network)
(Topic_1 implementation)

_5 (Topic_{11} single)
(Topic_{10} ocular dominance stripe)
(Topic_{12} low) (Topic_3 ocularity)
(Topic_{12} drift rate)

7 Finding the structure of proper names

Grammars offer structural and positional sensitivity that is not exploited in the basic LDA topic models. Here we explore the potential for using Bayesian inference for learning linear ordering constraints that hold between elements within proper names.

The Penn WSJ treebank is a widely used resource within computational linguistics (Marcus et al., 1993), but one of its weaknesses is that it does not indicate any structure internal to base noun phrases (i.e., it presents "flat" analyses of the pre-head NP elements). For many applications it would be extremely useful to have a more elaborated analysis of this kind of NP structure. For example, in an NP coreference application, if we could determine that *Bill* and *Hillary* are both first

[1] http://web.science.mq.edu.au/ ˜mjohnson/Software.htm

names then we could infer that *Bill Clinton* and *Hillary Clinton* are likely to refer to distinct individuals. On the other hand, because *Mr* in *Mr Clinton* is not a first name, it is possible that *Mr Clinton* and *Bill Clinton* refer to the same individual (Elsner et al., 2009).

Here we present an adaptor grammar based on the insights of the PCFG encoding of LDA topic models that learns some of the structure of proper names. The key idea is that elements in proper names typically appear in a fixed order; we expect honorifics to appear before first names, which appear before middle names, which in turn appear before surnames, etc. Similarly, many company names end in fixed phrases such as *Inc.* Here we think of first names as a kind of topic, albeit one with a restricted positional location. One of the challenges is that some of these structural elements can be filled by multiword expressions; e.g., *de Groot* can be a surname. We deal with this by permitting multi-word collocations to fill the corresponding positions, and use the adaptor grammar machinery to learn these collocations.

Inspired by the grammar presented in Elsner et al. (2009), our adaptor grammar is as follows, where adapted nonterminals are indicated by underlining as before.

$$
\begin{aligned}
&\mathrm{NP} \rightarrow (\mathrm{A0})\ (\mathrm{A1})\ \ldots\ (\mathrm{A6})\\
&\mathrm{NP} \rightarrow (\mathrm{B0})\ (\mathrm{B1})\ \ldots\ (\mathrm{B6})\\
&\mathrm{NP} \rightarrow \mathrm{Unordered}^{+}\\
&\underline{\mathrm{A0}} \rightarrow \mathrm{Word}^{+}\\
&\quad \ldots\\
&\underline{\mathrm{A6}} \rightarrow \mathrm{Word}^{+}\\
&\underline{\mathrm{B0}} \rightarrow \mathrm{Word}^{+}\\
&\quad \ldots\\
&\underline{\mathrm{B6}} \rightarrow \mathrm{Word}^{+}\\
&\underline{\mathrm{Unordered}} \rightarrow \mathrm{Word}^{+}
\end{aligned}
$$

In this grammar parentheses indicate optionality, and the Kleene plus indicates iteration (these were manually expanded into ordinary CFG rules in our experiments). The grammar provides three different expansions for proper names. The first expansion says that a proper name can consist of some subset of the six different collocation classes A0 through A6 in that order, while the second expansion says that a proper name can consist of some subset of the collocation classes B0 through B6, again in that order. Finally, the third expansion says that a proper name can consist of an arbitrary sequence of "unordered" collocations (this is intended as a "catch-all" expansion to provide analyses for proper names that don't fit either of the first two expansions).

We extracted all of the proper names (i.e., phrases of category NNP and NNPS) in the Penn WSJ treebank and used them as the training corpora for the adaptor grammar just described. The adaptor grammar inference procedure found skeletal sample parses such as the following:

(A0 barrett) (A3 smith)
(A0 albert) (A2 j.) (A3 smith) (A4 jr.)
(A0 robert) (A2 b.) (A3 van dover)
(B0 aim) (B1 prime rate) (B2 plus) (B5 fund) (B6 inc.)
(B0 balfour) (B1 maclaine) (B5 international) (B6 ltd.)
(B0 american express) (B1 information services) (B6 co)
(U abc) (U sports)
(U sports illustrated)
(U sports unlimited)

While a full evaluation will have to await further study, in general it seems to distinguish person names from company names reasonably reliably, and it seems to have discovered that person names consist of a first name (A0), a middle name or initial (A2), a surname (A3) and an optional suffix (A4). Similarly, it seems to have uncovered that company names typically end in a phrase such as *inc*, *ltd* or *co*.

8 Conclusion

This paper establishes a connection between two very different kinds of probabilistic models; LDA models of the kind used for topic modelling, and PCFGs, which are a standard model of hierarchical structure in language. The embedding we presented shows how to express an LDA model as a PCFG, and has the property that Bayesian inference of the parameters of that PCFG produces an equivalent model to that produced by Bayesian inference of the LDA model's parameters.

The primary value of this embedding is theoretical rather than practical; we are not advocating the use of PCFG estimation procedures to infer LDA models. Instead, we claim that the embedding suggests novel extensions to both the LDA topic models and PCFG-style grammars. We justified this claim by presenting several hybrid models that combine aspects of both topic models and

grammars. We don't claim that these are necessarily the best models for performing any particular tasks; rather, we present them as examples of models inspired by a combination of PCFGs and LDA topic models. We showed how the LDA to PCFG embedding suggested a "sticky topic" model extension to LDA. We then discussed adaptor grammars, and inspired by the LDA topic models, presented a novel topic model whose primitive elements are multi-word collocations rather than words. We concluded with an adaptor grammar that learns aspects of the internal structure of proper names.

Acknowledgments

This research was funded by US NSF awards 0544127 and 0631667, as well as by a start-up award from Macquarie University. I'd like to thank the organisers and audience at the Topic Modeling workshop at NIPS 2009, my former colleagues at Brown University (especially Eugene Charniak, Micha Elsner, Sharon Goldwater, Tom Griffiths and Erik Sudderth), my new colleagues at Macquarie University and the ACL reviewers for their excellent suggestions and comments on this work. Naturally all errors remain my own.

References

M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. 2002. The infinite Hidden Markov Model. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 577–584. The MIT Press.

N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jeffrey Elman. 1990. Finding structure in time. *Cognitive Science*, 14:197–211.

Micha Elsner, Eugene Charniak, and Mark Johnson. 2009. Structured generative models for unsupervised named-entity clustering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 164–172, Boulder, Colorado, June. Association for Computational Linguistics.

E. Fox, E. Sudderth, M. Jordan, and A. Willsky. 2008. An HDP-HMM for systems with state persistence. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 312–319. Omnipress.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466, Cambridge, MA. MIT Press.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:52285235.

Thomas L. Griffiths, Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211244.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

Mark Johnson. 2008. Using adaptor grammars to identifying synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio. Association for Computational Linguistics.

Kenichi Kurihara and Taisuke Sato. 2006. Variational Bayesian grammar induction for natural language. In *8th International Colloquium on Grammatical Inference*.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Probabilistic grammars and hierarchical Dirichlet processes. In *The Oxford Handbook of Applied Bayesian Analysis*. Oxford University Press.

Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Y. W. Teh, M. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM)*, pages 697–702.

C.S. Wetherell. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12:361–379.

A Cognitive Cost Model of Annotations Based on Eye-Tracking Data

Katrin Tomanek
Language & Information Engineering (JULIE) Lab
Universität Jena
Jena, Germany

Udo Hahn
Language & Information Engineering (JULIE) Lab
Universität Jena
Jena, Germany

Steffen Lohmann
Dept. of Computer Science & Applied Cognitive Science
Universität Duisburg-Essen
Duisburg, Germany

Jürgen Ziegler
Dept. of Computer Science & Applied Cognitive Science
Universität Duisburg-Essen
Duisburg, Germany

Abstract

We report on an experiment to track complex decision points in linguistic meta-data annotation where the decision behavior of annotators is observed with an eye-tracking device. As experimental conditions we investigate different forms of textual context and linguistic complexity classes relative to syntax and semantics. Our data renders evidence that annotation performance depends on the semantic and syntactic complexity of the decision points and, more interestingly, indicates that full-scale context is mostly negligible – with the exception of semantic high-complexity cases. We then induce from this observational data a cognitively grounded cost model of linguistic meta-data annotations and compare it with existing non-cognitive models. Our data reveals that the cognitively founded model explains annotation costs (expressed in annotation time) more adequately than non-cognitive ones.

1 Introduction

Today's NLP systems, in particular those relying on supervised ML approaches, are meta-data greedy. Accordingly, in the past years, we have witnessed a massive quantitative growth of annotated corpora. They differ in terms of the natural languages and domains being covered, the types of linguistic meta-data being solicited, and the text genres being served. We have seen large-scale efforts in syntactic and semantic annotations in the past related to POS tagging and parsing, on the one hand, and named entities and relations (propositions), on the other hand. More recently, we are dealing with even more challenging issues such as subjective language, a large variety of co-reference and (e.g., RST-style) text structure phenomena, Since the NLP community is further extending their work into these more and more sophisticated semantic and pragmatic analytics, there seems to be no end in sight for increasingly complex and diverse annotation tasks.

Yet, producing annotations is pretty expensive. So the question comes up, how we can rationally manage these investments so that annotation campaigns are economically doable without loss in annotation quality. The economics of annotations are at the core of *Active Learning* (AL) where those linguistic samples are focused on in the entire document collection, which are estimated as being most informative to learn an effective classification model (Cohn et al., 1996). This intentional selection bias stands in stark contrast to prevailing sampling approaches where annotation examples are randomly chosen.

When different approaches to AL are compared with each other, or with standard random sampling, in terms of annotation efficiency, up until now, the AL community assumed *uniform* annotation costs for each linguistic unit, e.g. words. This claim, however, has been shown to be invalid in several studies (Hachey et al., 2005; Settles et al., 2008; Tomanek and Hahn, 2010). If uniformity does not hold and, hence, the number of annotated units does not indicate the true annotation efforts required for a specific sample, empirically more adequate cost models are needed.

Building predictive models for annotation costs has only been addressed in few studies for now (Ringger et al., 2008; Settles et al., 2008; Arora et al., 2009). The proposed models are based on easy-to-determine, yet not so explanatory variables (such as the number of words to be annotated), indicating that accurate models of annotation costs remain a desideratum. We here, alternatively, consider different classes of syntactic and semantic complexity that might affect the cognitive load during the annotation process, with

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1158–1167,
Uppsala, Sweden, 11-16 July 2010.

the overall goal to find additional and empirically more adequate variables for cost modeling.

The complexity of linguistic utterances can be judged either by structural or by behavioral criteria. Structural complexity emerges, e.g., from the static topology of phrase structure trees and procedural graph traversals exploiting the topology of parse trees (see Szmrecsányi (2004) or Cheung and Kemper (1992) for a survey of metrics of this type). However, structural complexity criteria do not translate directly into empirically justified cost measures and thus have to be taken with care.

The behavioral approach accounts for this problem as it renders observational data of the annotators' eye movements. The technical vehicle to gather such data are eye-trackers which have already been used in psycholinguistics (Rayner, 1998). Eye-trackers were able to reveal, e.g., how subjects deal with ambiguities (Frazier and Rayner, 1987; Rayner et al., 2006; Traxler and Frazier, 2008) or with sentences which require re-analysis, so-called garden path sentences (Altmann et al., 2007; Sturt, 2007).

The rationale behind the use of eye-tracking devices for the observation of annotation behavior is that the length of gaze durations and behavioral patterns underlying gaze movements are considered to be indicative of the hardness of the linguistic analysis and the expenditures for the search of clarifying linguistic evidence (anchor words) to resolve hard decision tasks such as phrase attachments or word sense disambiguation. Gaze duration and search time are then taken as empirical correlates of linguistic complexity and, hence, uncover the *real* costs. We therefore consider eye-tracking as a promising means to get a better understanding of the nature of the linguistic annotation processes with the ultimate goal of identifying predictive factors for annotation cost models.

In this paper, we first describe an empirical study where we observed the annotators' reading behavior while annotating a corpus. Section 2 deals with the design of the study, Section 3 discusses its results. In Section 4 we then focus on the implications this study has on building cost models and compare a simple cost model mainly relying on word and character counts and additional simple descriptive characteristics with one that can be derived from experimental data as provided from eye-tracking. We conclude with experiments which reveal that cognitively grounded models outperform simpler ones relative to cost prediction using annotation time as a cost measure. Based on this finding, we suggest that cognitive criteria are helpful for uncovering the real costs of corpus annotation.

2 Experimental Design

In our study, we applied, for the first time ever to the best of our knowledge, eye-tracking to study the cognitive processes underlying the annotation of linguistic meta-data, named entities in particular. In this task, a human annotator has to decide for each word whether or not it belongs to one of the entity types of interest.

We used the English part of the MUC7 corpus (Linguistic Data Consortium, 2001) for our study. It contains *New York Times* articles from 1996 reporting on plane crashes. These articles come already annotated with three types of named entities considered important in the newspaper domain, *viz.* "persons", "locations", and "organizations".

Annotation of these entity types in newspaper articles is admittedly fairly easy. We chose this rather simple setting because the participants in the experiment had no previous experience with document annotation and no serious linguistic background. Moreover, the limited number of entity types reduced the amount of participants' training prior to the actual experiment, and positively affected the design and handling of the experimental apparatus (see below).

We triggered the annotation processes by giving our participants specific *annotation examples*. An example consists of a text document having one single *annotation phrase* highlighted which then had to be semantically annotated with respect to named entity mentions. The annotation task was defined such that the correct entity type had to be assigned to each word in the annotation phrase. If a word belongs to none of the three entity types a fourth class called "no entity" had to be assigned.

The phrases highlighted for annotation were *complex noun phrases* (CNPs), each a sequence of words where a noun (or an equivalent nominal expression) constitutes the syntactic head and thus dominates dependent words such as determiners, adjectives, or other nouns or nominal expressions (including noun phrases and prepositional phrases). CNPs with even more elaborate internal syntactic structures, such as coordinations, appositions, or relative clauses, were isolated from

their syntactic host structure and the intervening linguistic material containing these structures was deleted to simplify overly long sentences. We also discarded all CNPs that did not contain at least one *entity-critical* word, i.e., one which might be a named entity according to its orthographic appearance (e.g., starting with an upper-case letter). It should be noted that such orthographic signals are by no means a sufficient condition for the presence of a named entity mention within a CNP.

The choice of CNPs as stimulus phrases is motivated by the fact that named entities are usually fully encoded by this kind of linguistic structure. The chosen stimulus – an annotation example with one phrase highlighted for annotation – allows for an exact localization of the cognitive processes and annotation actions performed relative to that specific phrase.

2.1 Independent Variables

We defined two measures for the complexity of the annotation examples: The *syntactic* complexity was given by the number of nodes in the constituent parse tree which are dominated by the annotation phrase (Szmrecsányi, 2004).[1] According to a threshold on the number of nodes in such a parse tree, we classified CNPs as having either high or low syntactic complexity.

The *semantic* complexity of an annotation example is based on the inverse document frequency df of the words in the annotation phrase according to a reference corpus.[2] We calculated the semantic complexity score of an annotation phrase as $max_i \frac{1}{df(w_i)}$, where w_i is the i-th word of the annotation phrase. Again, we empirically determined a threshold classifying annotation phrases as having either high or low semantic complexity. Additionally, this automatically generated classification was manually checked and, if necessary, revised by two annotation experts. For instance, if an annotation phrase contained a strong trigger (e.g., a social role or job title, as with *"spokeswoman"* in the annotation phrase *"spokeswoman Arlene"*), it was classified as a low-semantic-complexity item even though it might have been assigned a high inverse document frequency (due to the infrequent word *"Arlene"*).

[1]Constituency parse structure was obtained from the OpenNLP parser (`http://opennlp.sourceforge.net/`) trained on PennTreeBank data.

[2]We chose the English part of the Reuters RCV2 corpus as the reference corpus for our experiments.

Two experimental groups were formed to study different contexts. In the *document context* condition the whole newspaper article was shown as annotation example, while in the *sentence context* condition only the sentence containing the annotation phrase was presented. The participants[3] were randomly assigned to one of these groups. We decided for this between-subjects design to avoid any irritation of the participants caused by constantly changing contexts. Accordingly, the participants were assigned to one of the experimental groups and corresponding context condition already in the second training phase that took place shortly before the experiment started (see below).

2.2 Hypotheses and Dependent Variables

We tested the following two hypotheses:

Hypothesis H1: *Annotators perform differently in the two context conditions.*

H1 is based on the linguistically plausible assumption that annotators are expected to make heavy use of the surrounding context because such context could be helpful for the correct disambiguation of entity classes. Accordingly, lacking context, an annotator is expected to annotate worse than under the condition of full context. However, the availability of (too much) context might overload and distract annotators, with a presumably negative effect on annotation performance.

Hypothesis H2: *The complexity of the annotation phrases determines the annotation performance.*

The assumption is that high syntactic or semantic complexity significantly lowers the annotation performance.

In order to test these hypotheses we collected data for the following dependent variables: *(a)* the annotation accuracy – we identified erroneous entities by comparison with the original gold annotations in the MUC7 corpus, *(b)* the time needed per annotation example, and *(c)* the distribution and duration of the participants' eye gazes.

[3]20 subjects (12 female) with an average age of 24 years (mean = 24, standard deviation (SD) = 2.8) and normal or corrected-to-normal vision capabilities took part in the study. All participants were students with a computing-related study background, with good to very good English language skills (mean = 7.9, SD = 1.2, on a ten-point scale with 1 = "poor" and 10 = "excellent", self-assessed), but without any prior experience in annotation and without previous exposure to linguistic training.

2.3 Stimulus Material

According to the above definition of complexity, we automatically preselected annotation examples characterized by either a low or a high degree of semantic and syntactic complexity. After manual fine-tuning of the example set assuring an even distribution of entity types and syntactic correctness of the automatically derived annotation phrases, we finally selected 80 annotation examples for the experiment. These were divided into four subsets of 20 examples each falling into one of the following complexity classes:

sem-syn: low semantic/low syntactic complexity
SEM-syn: high semantic/low syntactic complexity
sem-SYN: low semantic/high syntactic complexity
SEM-SYN: high semantic/high syntactic complexity

2.4 Experimental Apparatus and Procedure

The annotation examples were presented in a custom-built tool and its user interface was kept as simple as possible not to distract the eye movements of the participants. It merely contained one frame showing the text of the annotation example, with the annotation phrase being highlighted. A blank screen was shown after each annotation example to reset the eyes and to allow a break, if needed. The time the blank screen was shown was not counted as annotation time. The 80 annotation examples were presented to all participants in the same randomized order, with a balanced distribution of the complexity classes. A variation of the order was hardly possible for technical and analytical reasons but is not considered critical due to extensive, pre-experimental training (see below). The limitation on 80 annotation examples reduces the chances of errors due to fatigue or lack of attention that can be observed in long-lasting annotation activities.

Five introductory examples (not considered in the final evaluation) were given to get the subjects used to the experimental environment. All annotation examples were chosen in a way that they completely fitted on the screen (i.e., text length was limited) to avoid the need for scrolling (and eye distraction). The position of the CNP within the respective context was randomly distributed, excluding the first and last sentence.

The participants used a standard keyboard to assign the entity types for each word of the annotation example. All but 5 keys were removed from the keyboard to avoid extra eye movements for finger coordination (three keys for the positive entity classes, one for the negative "no entity" class, and one to confirm the annotation). Pre-tests had shown that the participants could easily issue the annotations without looking down at the keyboard.

We recorded the participant's eye movements on a Tobii T60 eye-tracking device which is invisibly embedded in a 17" TFT monitor and comparatively tolerant to head movements. The participants were seated in a comfortable position with their head in a distance of 60-70 cm from the monitor. Screen resolution was set to 1280 x 1024 px and the annotation examples were presented in the middle of the screen in a font size of 16 px and a line spacing of 5 px. The presentation area had no fixed height and varied depending on the context condition and length of the newspaper article. The text was always vertically centered on the screen.

All participants were familiarized with the annotation task and the guidelines in a pre-experimental workshop where they practiced annotations on various exercise examples (about 60 minutes). During the next two days, one after the other participated in the actual experiment which took between 15 and 30 minutes, including calibration of the eye-tracking device. Another 20-30 minutes of training time directly preceded the experiment. After the experiment, participants were interviewed and asked to fill out a questionnaire. Overall, the experiment took about two hours for each participant for which they were financially compensated. Participants were instructed to focus more on annotation accuracy than on annotation time as we wanted to avoid random guessing. Accordingly, as an extra incentive, we rewarded the three participants with the highest annotation accuracy with cinema vouchers. None of the participants reported serious difficulties with the newspaper articles or annotation tool and all understood the annotation task very well.

3 Results

We used a mixed-design analysis of variance (ANOVA) model to test the hypotheses, with the context condition as between-subjects factor and the two complexity classes as within-subject factors.

3.1 Testing Context Conditions

To test hypothesis H1 we compared the number of annotation errors on entity-critical words made

	above	before	anno phrase	after	below
percentage of participants looking at a sub-area	35%	32%	100%	34%	16%
average number of fixations per sub-area	2.2		14.1		1.3

Table 1: Distribution of annotators' attention among sub-areas per annotation example.

by the annotators in the two contextual conditions (complete document *vs.* sentence). Surprisingly, on the total of 174 entity-critical words within the 80 annotation examples, we found exactly the same mean value of 30.8 errors per participant in both conditions. There were also no significant differences in the average time needed to annotate an example in both conditions (means of 9.2 and 8.6 seconds, respectively, with $F(1, 18) = 0.116$, $p = 0.74$).[4] These results seem to suggest that it makes no difference (neither for annotation accuracy nor for time) whether or not annotators are shown textual context beyond the sentence that contains the annotation phrase.

To further investigate this finding we analyzed eye-tracking data of the participants gathered for the document context condition. We divided the whole text area into five sub-areas as schematically shown in Figure 1. We then determined the average proportion of participants that directed their gaze at least once at these sub-areas. We considered all fixations with a minimum duration of 100 ms, using a fixation radius (i.e., the smallest distance that separates fixations) of 30 px and excluded the first second (mainly used for orientation and identification of the annotation phrase).

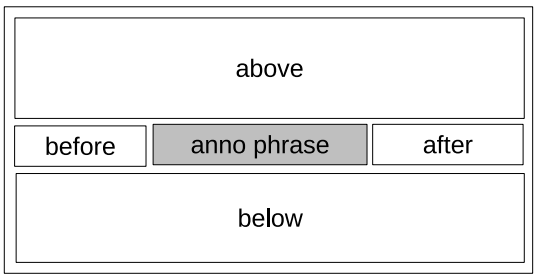


Figure 1: Schematic visualization of the sub-areas of an annotation example.

Table 1 reveals that on average only 35% of the participants looked in the textual context above the annotation phrase embedding sentence, and even less perceived the context below (16%). The sentence parts before and after the annotation phrase were, on the average, visited by one third (32% and 34%, respectively) of the participants. The uneven distribution of the annotators' attention becomes even more apparent in a comparison of the total number of fixations on the different text parts: 14 out of an average of 18 fixations per example were directed at the annotation phrase and the surrounding sentence, the text context above the annotation chunk received only 2.2 fixations on the average and the text context below only 1.3.

Thus, the eye-tracking data indicates that the textual context is not as important as might have been expected for quick and accurate annotation. This result can be explained by the fact that participants of the document-context condition used the context whenever they thought it might help, whereas participants of the sentence-context condition spent more time thinking about a correct answer, overall with the same result.

3.2 Testing Complexity Classes

To test hypothesis H2 we also compared the average annotation time and the number of errors on entity-critical words for the complexity subsets (see Table 2). The ANOVA results show highly significant differences for both annotation time and errors.[5] A pairwise comparison of all subsets in both conditions with a t-test showed non-significant results only between the SEM-syn and syn-SEM subsets.[6]

Thus, the empirical data generally supports hypothesis H2 in that the annotation performance seems to correlate with the complexity of the annotation phrase, on the average.

[4] In general, we observed a high variance in the number of errors and time values between the subjects. While, e.g., the fastest participant handled an example in 3.6 seconds on the average, the slowest one needed 18.9 seconds; concerning the annotation errors on the 174 entity-critical words, these ranged between 21 and 46 errors.

[5] Annotation time results: $F(1, 18) = 25$, $p < 0.01$ for the semantic complexity and $F(1, 18) = 76.5$, $p < 0.01$ for the syntactic complexity; Annotation complexity results: $F(1, 18) = 48.7$, $p < 0.01$ for the semantic complexity and $F(1, 18) = 184$, $p < 0.01$ for the syntactic complexity.

[6] $t(9) = 0.27$, $p = 0.79$ for the annotation time in the document context condition, and $t(9) = 1.97$, $p = 0.08$ for the annotation errors in the sentence context condition.

experimental condition	complexity class	e.-c. words	time mean	time SD	errors mean	errors SD	errors rate
document condition	sem-syn	36	4.0s	2.0	2.7	2.1	.075
	SEM-syn	25	9.2s	6.7	5.1	1.4	.204
	sem-SYN	51	9.6s	4.0	9.1	2.9	.178
	SEM-SYN	62	14.2s	9.5	13.9	4.5	.224
sentence condition	sem-syn	36	3.9s	1.3	1.1	1.4	.031
	SEM-syn	25	7.5s	2.8	6.2	1.9	.248
	sem-SYN	51	9.6s	2.8	9.0	3.9	.176
	SEM-SYN	62	13.5s	5.0	14.5	3.4	.234

Table 2: Average performance values for the 10 subjects of each experimental condition and 20 annotation examples of each complexity class: number of entity-critical words, mean annotation time and standard deviations (SD), mean annotation errors, standard deviations, and error rates (number of errors divided by number of entity-critical words).

3.3 Context and Complexity

We also examined whether the need for inspecting the context increases with the complexity of the annotation phrase. Therefore, we analyzed the eye-tracking data in terms of the average number of fixations on the annotation phrase and on its embedding contexts for each complexity class (see Table 3). The values illustrate that while the number of fixations on the annotation phrase rises generally with both the semantic and the syntactic complexity, the number of fixations on the context rises only with semantic complexity. The number of fixations on the context is nearly the same for the two subsets with low semantic complexity (sem-syn and sem-SYN, with 1.0 and 1.5), while it is significantly higher for the two subsets with high semantic complexity (5.6 and 5.0), independent of the syntactic complexity.[7]

complexity class	fix. on phrase mean	fix. on phrase SD	fix. on context mean	fix. on context SD
sem-syn	4.9	4.0	1.0	2.9
SEM-syn	8.1	5.4	5.6	5.6
sem-SYN	18.1	7.7	1.5	2.0
SEM-SYN	25.4	9.3	5.0	4.1

Table 3: Average number of fixations on the annotation phrase and context for the document condition and 20 annotation examples of each complexity class.

These results suggest that the need for context mainly depends on the semantic complexity of the annotation phrase, while it is less influenced by its syntactic complexity.

[7] ANOVA result of $F(1, 19) = 19.7$, $p < 0.01$ and significant differences also in all pairwise comparisons.

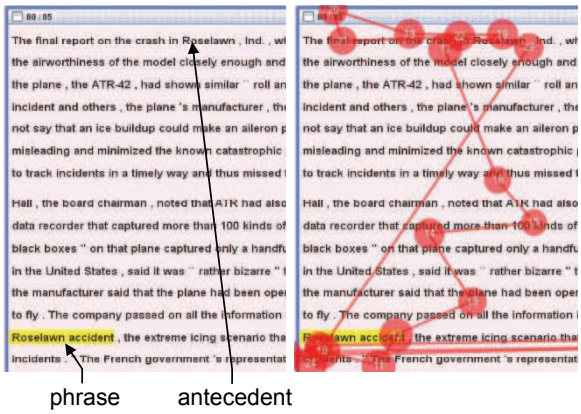


Figure 2: Annotation example with annotation phrase and the antecedent for *"Roselawn"* in the text (left), and gaze plot of one participant showing a scanning-for-coreference behavior (right).

This finding is also qualitatively supported by the gaze plots we generated from the eye-tracking data. Figure 2 shows a gaze plot for one participant that illustrates a scanning-for-coreference behavior we observed for several annotation phrases with high semantic complexity. In this case, words were searched in the upper context, which according to their orthographic signals might refer to a named entity but which could not completely be resolved only relying on the information given by the annotation phrase itself and its embedding sentence. This is the case for *"Roselawn"* in the annotation phrase *"Roselawn accident"*. The context reveals that Roselawn, which also occurs in the first sentence, is a location. A similar procedure is performed for acronyms and abbreviations which cannot be resolved from the immediate local context – searches mainly visit the upper context. As indicated by the gaze movements, it also became apparent that texts were rather scanned for hints instead of being deeply read.

4 Cognitively Grounded Cost Modeling

We now discuss whether the findings on dependent variables from our eye-tracking study are fruitful for actually modeling annotation costs. Therefore, we learn a linear regression model with time (an operationalization of annotation costs) as the dependent variable. We compare our 'cognitive' model against a baseline model which relies on some simple formal text features only, and test whether the newly introduced features help predict annotation costs more accurately.

4.1 Features

The features for the baseline model, character- and word-based, are similar to the ones used by Ringger et al. (2008) and Settles et al. (2008).[8] Our cognitive model, however, makes additional use of features based on linguistic complexity, and includes syntactic and semantic criteria related to the annotation phrases. These features were inspired by the insights provided by our eye-tracking experiments. All features are designed such that they can automatically be derived from *unlabeled* data, a necessary condition for such features to be practically applicable.

To account for our findings that syntactic and semantic complexity correlates with annotation performance, we added three features based on syntactic, and two based on semantic complexity measures. We decided for the use of multiple measures because there is no single agreed-upon metric for either syntactic or semantic complexity. This decision is further motivated by findings which reveal that different measures are often complementary to each other so that their combination better approximates the inherent degrees of complexity (Roark et al., 2007).

As for syntactic complexity, we use two measures based on structural complexity including *(a)* the number of nodes of a constituency parse tree which are dominated by the annotation phrase (cf. Section 2.1), and *(b)* given the dependency graph of the sentence embedding the annotation phrase, we consider the distance between words for each dependency link within the annotation phrase and consider the maximum over such distance values as another metric for syntactic complexity. Lin (1996) has already shown that human performance on sentence processing tasks can be predicted using such a measure. Our third syntactic complexity measure is based on the probability of part-of-speech (POS) 2-grams. Given a POS 2-gram model, which we learned from the automatically POS-tagged MUC7 corpus, the complexity of an annotation phrase is defined by $\sum_{i=2}^{n} P(\text{POS}_i|\text{POS}_{i-1})$ where POS_i refers to the POS-tag of the i-th word of the annotation phrase. A similar measure has been used by Roark et al. (2007) who claim that complex syntactic structures correlate with infrequent or surprising combinations of POS tags.

As far as the quantification of semantic complexity is concerned, we use *(a)* the inverse document frequency $df(w_i)$ of each word w_i (cf. Section 2.1), and a measure based on the semantic ambiguity of each word, i.e., the number of meanings contained in WORDNET,[9] within an annotation phrase. We consider the maximum ambiguity of the words within the annotation phrase as the overall ambiguity of the respective annotation phrase. This measure is based on the assumption that annotation phrases with higher semantic ambiguity are harder to annotate than low-ambiguity ones. Finally, we add the Flesch-Kincaid Readability Score (Klare, 1963), a well-known metric for estimating the comprehensibility and reading complexity of texts.

As already indicated, some of the hardness of annotations is due to tracking co-references and abbreviations. Both often cannot be resolved locally so that annotators need to consult the context of an annotation chunk (cf. Section 3.3). Thus, we also added features providing information whether the annotation phrases contain entity-critical words which may denote the referent of an antecedent of an anaphoric relation. In the same vein, we checked whether an annotation phrase contains expressions which can function as an abbreviation by virtue of their orthographical appearance, e.g., consist of at least two upper-case letters.

Since our participants were sometimes scanning for entity-critical words, we also added features providing information on the number of entity-critical words within the annotation phrase. Table 4 enumerates all feature classes and single features used for determining our cost model.

[8] In preliminary experiments our set of basic features comprised additional features providing information on the usage of stop words in the annotation phrase and on the number of paragraphs, sentences, and words in the respective annotation example. However, since we found these features did not have any significant impact on the model, we removed them.

[9] `http://wordnet.princeton.edu/`

Feature Group	# Features	Feature Description
characters (basic)	6	number of characters and words per annotation phrase; test whether words in a phrase start with capital letters, consist of capital letters only, have alphanumeric characters, or are punctuation symbols
words	2	number of entity-critical words and percentage of entity-critical words in the annotation phrase
complexity	6	syntactic complexity: number of dominated nodes, POS n-gram probability, maximum dependency distance; semantic complexity: inverse document frequency, max. ambiguity; general linguistic complexity: Flesch-Kincaid Readability Score
semantics	3	test whether entity-critical word in annotation phrase is used in document (preceding or following current phrase); test whether phrase contains an abbreviation

Table 4: Features for cost modeling.

4.2 Evaluation

To test how well annotation costs can be modeled by the features described above, we used the MUC7$_\mathcal{T}$ corpus, a re-annotation of the MUC7 corpus (Tomanek and Hahn, 2010). MUC7$_\mathcal{T}$ has time tags attached to the sentences and CNPs. These time tags indicate the time it took to annotate the respective phrase for named entity mentions of the types *person*, *location*, and *organization*. We here made use of the time tags of the 15,203 CNPs in MUC7$_\mathcal{T}$. MUC7$_\mathcal{T}$ has been annotated by two annotators (henceforth called *A* and *B*) and so we evaluated the cost models for both annotators. We learned a simple linear regression model with the annotation time as dependent variable and the features described above as independent variables. The baseline model only includes the basic feature set, whereas the 'cognitive' model incorporates all features described above.

Table 5 depicts the performance of both models induced from the data of annotator A and B. The coefficient of determination (R^2) describes the proportion of the variance of the dependent variable that can be described by the given model. We report adjusted R^2 to account for the different numbers of features used in both models.

model	R^2 on A's data	R^2 on B's data
baseline	0.4695	0.4640
cognitive	0.6263	0.6185

Table 5: Adjusted R^2 values on both models and for annotators A and B.

For both annotators, the baseline model is significantly outperformed in terms of R^2 by our 'cognitive' model ($p < 0.05$). Considering the features that were inspired from the eye-tracking study, R^2 is increased from 0.4695 to 0.6263 on the timing data of annotator A, and from 0.464 to 0.6185 on the data of annotator B. These numbers clearly demonstrate that annotation costs are more adequately modelled by the additional features we identified through our eye-tracking study.

Our 'cognitive' model now consists of 21 coefficients. We tested for the significance of this model's regression terms. For annotator A we found all coefficients to be significant with respect to the model ($p < 0.05$), for annotator B all coefficients except one were significant. Figure 6 shows the coefficients of annotator A's 'cognitive' model along with the standard errors and t-values.

5 Summary and Conclusions

In this paper, we explored the use of eye-tracking technology to investigate the behavior of human annotators during the assignment of three types of named entities – persons, organizations and locations – based on the eye-mind assumption. We tested two main hypotheses – one relating to the amount of contextual information being used for annotation decisions, the other relating to different degrees of syntactic and semantic complexity of expressions that had to be annotated. We found experimental evidence that the textual context is searched for decision making on assigning semantic meta-data at a surprisingly low rate (with

Feature Group	Feature Name/Coefficient	Estimate	Std. Error	t value	Pr(>|t|)
	(Intercept)	855.0817	33.3614	25.63	0.0000
characters (basic)	token_number	-304.3241	29.6378	-10.27	0.0000
	char_number	7.1365	2.2622	3.15	0.0016
	has_token_initcaps	244.4335	36.1489	6.76	0.0000
	has_token_allcaps	-342.0463	62.3226	-5.49	0.0000
	has_token_alphanumeric	-197.7383	39.0354	-5.07	0.0000
	has_token_punctuation	-303.7960	50.3570	-6.03	0.0000
words	number_tokens_entity_like	934.3953	13.3058	70.22	0.0000
	percentage_tokens_entity_like	-729.3439	43.7252	-16.68	0.0000
complexity	sem_compl_inverse_document_freq	392.8855	35.7576	10.99	0.0000
	sem_compl_maximum_ambiguity	-13.1344	1.8352	-7.16	0.0000
	synt_compl_number_dominated_nodes	87.8573	7.9094	11.11	0.0000
	synt_compl_pos_ngram_probability	287.8137	28.2793	10.18	0.0000
	syn_complexity_max_dependency_distance	28.7994	9.2174	3.12	0.0018
	flesch_kincaid_readability	-0.4117	0.1577	-2.61	0.0090
semantics	has_entity_critical_token_used_above	73.5095	24.1225	3.05	0.0023
	has_entity_critical_token_used_below	-178.0314	24.3139	-7.32	0.0000
	has_abbreviation	763.8605	73.5328	10.39	0.0000

Table 6: 'Cognitive' model of annotator A.

the exception of tackling high-complexity semantic cases and resolving co-references) and that annotation performance correlates with semantic and syntactic complexity.

The results of these experiments were taken as a heuristic clue to focus on cognitively plausible features of learning empirically rooted cost models for annotation. We compared a simple cost model (basically taking the number of words and characters into account) with a cognitively grounded model and got a much higher fit for the cognitive model when we compared cost predictions of both model classes on the recently released time-stamped version of the MUC7 corpus.

We here want to stress the role of cognitive evidence from eye-tracking to determine *empirically relevant* features for the cost model. The alternative, more or less mechanical feature engineering, suffers from the shortcoming that is has to deal with large amounts of (mostly irrelevant) features – a procedure which not only requires increased amounts of training data but also is often computationally very expensive.

Instead, our approach introduces empirical, theory-driven relevance criteria into the feature selection process. Trying to relate observables of complex cognitive tasks (such as gaze duration and gaze movements for named entity annotation) to explanatory models (in our case, a time-based cost model for annotation) follows a much warranted avenue in research in NLP where feature farming becomes a theory-driven, explanatory process rather than a much deplored theory-blind engineering activity (cf. ACL-WS-2005 (2005)).

In this spirit, our focus has not been on fine-tuning this cognitive cost model to achieve even higher fits with the time data. Instead, we aimed at testing whether the findings from our eye-tracking study can be exploited to model annotation costs more accurately.

Still, future work will be required to optimize a cost model for eventual application where even more accurate cost models may be required. This optimization may include both exploration of additional features (such as domain-specific ones) as well as experimentation with other, presumably non-linear, regression models. Moreover, the impact of improved cost models on the efficiency of (cost-sensitive) selective sampling approaches, such as Active Learning (Tomanek and Hahn, 2009), should be studied.

References

ACL-WS-2005. 2005. *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*. accessible via `http://www.aclweb.org/anthology/W/W05/W05-0400.pdf`.

Gerry Altmann, Alan Garnham, and Yvette Dennis. 2007. Avoiding the garden path: Eye movements in context. *Journal of Memory and Language*, 31(2):685–712.

Shilpa Arora, Eric Nyberg, and Carolyn Rosé. 2009. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 18–26.

Hintat Cheung and Susan Kemper. 1992. Competing complexity metrics and adults' production of complex sentences. *Applied Psycholinguistics*, 13:53–76.

David Cohn, Zoubin Ghahramani, and Michael Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Lyn Frazier and Keith Rayner. 1987. Resolution of syntactic category ambiguities: Eye movements in parsing lexically ambiguous sentences. *Journal of Memory and Language*, 26:505–526.

Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *CoNLL 2005 – Proceedings of the 9th Conference on Computational Natural Language Learning*, pages 144–151.

George Klare. 1963. *The Measurement of Readability*. Ames: Iowa State University Press.

Dekang Lin. 1996. On the structural complexity of natural language sentences. In *COLING 1996 – Proceedings of the 16th International Conference on Computational Linguistics*, pages 729–733.

Linguistic Data Consortium. 2001. Message Understanding Conference (MUC) 7. Philadelphia: Linguistic Data Consortium.

Keith Rayner, Anne Cook, Barbara Juhasz, and Lyn Frazier. 2006. Immediate disambiguation of lexically ambiguous words during reading: Evidence from eye movements. *British Journal of Psychology*, 97:467–482.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 126:372–422.

Eric Ringger, Marc Carmen, Robbie Haertel, Kevin Seppi, Deryle Lonsdale, Peter McClanahan, James Carroll, and Noel Ellison. 2008. Assessing the costs of machine-assisted corpus annotation through a user study. In *LREC 2008 – Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 3318–3324.

Brian Roark, Margaret Mitchell, and Kristy Hollingshead. 2007. Syntactic complexity measures for detecting mild cognitive impairment. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 1–8.

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS 2008 Workshop on Cost-Sensitive Machine Learning*, pages 1–10.

Patrick Sturt. 2007. Semantic re-interpretation and garden path recovery. *Cognition*, 105:477–488.

Benedikt M. Szmrecsányi. 2004. On operationalizing syntactic complexity. In *Proceedings of the 7th International Conference on Textual Data Statistical Analysis. Vol. II*, pages 1032–1039.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *ACL 2009 – Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 1039–1047.

Katrin Tomanek and Udo Hahn. 2010. Annotation time stamps: Temporal metadata from the linguistic annotation process. In *LREC 2010 – Proceedings of the 7th International Conference on Language Resources and Evaluation*.

Matthew Traxler and Lyn Frazier. 2008. The role of pragmatic principles in resolving attachment ambiguities: Evidence from eye movements. *Memory & Cognition*, 36:314–328.

A Rational Model of Eye Movement Control in Reading

Klinton Bicknell and **Roger Levy**
Department of Linguistics
University of California, San Diego
9500 Gilman Dr, La Jolla, CA 92093-0108
{kbicknell,rlevy}@ling.ucsd.edu

Abstract

A number of results in the study of real-time sentence comprehension have been explained by computational models as resulting from the rational use of probabilistic linguistic information. Many times, these hypotheses have been tested in reading by linking predictions about relative word difficulty to word-aggregated eye tracking measures such as go-past time. In this paper, we extend these results by asking to what extent reading is well-modeled as rational behavior at a finer level of analysis, predicting not aggregate measures, but the duration and location of each fixation. We present a new rational model of eye movement control in reading, the central assumption of which is that eye movement decisions are made to obtain noisy visual information as the reader performs Bayesian inference on the identities of the words in the sentence. As a case study, we present two simulations demonstrating that the model gives a rational explanation for between-word regressions.

1 Introduction

The language processing tasks of reading, listening, and even speaking are remarkably difficult. Good performance at each one requires integrating a range of types of probabilistic information and making incremental predictions on the basis of noisy, incomplete input. Despite these requirements, empirical work has shown that humans perform very well (e.g., Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995). Sophisticated models have been developed that explain many of these effects using the tools of computational linguistics and large-scale corpora to make normative predictions for optimal performance in these tasks (Genzel & Charniak, 2002, 2003; Keller, 2004; Levy & Jaeger, 2007; Jaeger, 2010). To the extent that the behavior of these models looks like human behavior, it suggests that humans are making rational use of all the information available to them in language processing. In the domain of incremental language comprehension, especially, there is a substantial amount of computational work suggesting that humans behave rationally (e.g., Jurafsky, 1996; Narayanan & Jurafsky, 2001; Levy, 2008; Levy, Reali, & Griffiths, 2009). Most of this work has taken as its task predicting the difficulty of each word in a sentence, a major result being that a large component of the difficulty of a word appears to be a function of its probability in context (Hale, 2001; Smith & Levy, 2008). Much of the empirical basis for this work comes from studying reading, where word difficulty can be related to the amount of time that a reader spends on a particular word. To relate these predictions about word difficulty to the data obtained in eye tracking experiments, the eye movement record has been summarized through word aggregate measures, such as the average duration of the first fixation on a word, or the amount of time between when a word is first fixated and when the eyes move to its right ('go-past time').

It is important to note that this notion of word difficulty is an abstraction over the actual task of reading, which is made up of more fine-grained decisions about how long to leave the eyes in their current position, and where to move them next, producing the series of relatively stable periods (fixations) and movements (saccades) that characterize the eye tracking record. While there has been much empirical work on reading at this fine-grained scale (see Rayner, 1998 for an overview), and there are a number of successful models (Reichle, Pollatsek, & Rayner, 2006; Engbert, Nuthmann, Richter, & Kliegl, 2005), little is known about the extent to which human reading behavior appears to be rational at this finer

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1168–1178,
Uppsala, Sweden, 11-16 July 2010.

grained scale. In this paper, we present a new rational model of eye movement control in reading, the central assumption of which is that eye movement decisions are made to obtain noisy visual information, which the reader uses in Bayesian inference about the form and structure of the sentence. As a case study, we show that this model gives a rational explanation for between-word regressions.

In Section 2, we briefly describe the leading models of eye movements in reading, and in Section 3, we describe how these models account for between-word regressions and the intuition behind our model's account of them. Section 4 describes the model and its implementation and Sections 5–6 describe two simulations we performed with the model comparing behavioral policies that make regressions to those that do not. In Simulation 1, we show that specific regressive policies outperform specific non-regressive policies, and in Simulation 2, we use optimization to directly find optimal policies for three performance measures. The results show that the regressive policies outperform non-regressive policies across a wide range of performance measures, demonstrating that our model predicts that making between-word regressions is a rational strategy for reading.

2 Models of eye movements in reading

The two most successful models of eye movements in reading are *E-Z* Reader (Reichle, Pollatsek, Fisher, & Rayner, 1998; Reichle et al., 2006) and SWIFT (Engbert, Longtin, & Kliegl, 2002; Engbert et al., 2005). Both of these models characterize the problem of reading as one of word identification. In *E-Z* Reader, for example, the system identifies each word in the sentence serially, moving attention to the next word in the sentence only after processing the current word is complete, and (to slightly oversimplify), the eyes then follow the attentional shifts at some lag. SWIFT works similarly, but with the main difference being that processing and attention are distributed over multiple words, such that adjacent words can be identified in parallel. While both of these models provide a good fit to eye tracking data from reading, neither model asks the higher level question of what a rational solution to the problem would look like.

The first model to ask this question, Mr. Chips (Legge, Klitz, & Tjan, 1997; Legge, Hooven, Klitz, Mansfield, & Tjan, 2002), predicts the optimal sequence of saccade targets to read a text based on a principle of minimizing the expected entropy in the distribution over identities of the current word. Unfortunately, however, the Mr. Chips model simplifies the problem of reading in a number of ways: First, it uses a unigram model as its language model, and thus fails to use any information in the linguistic context to help with word identification. Second, it only moves on to the next word after unambiguous identification of the current word, whereas there is experimental evidence that comprehenders maintain some uncertainty about the word identities. In other work, we have extended the Mr. Chips model to remove these two limitations, and show that the resulting model more closely matches human performance (Bicknell & Levy, 2010). The larger problem, however, is that each of these models uses an unrealistic model of visual input, which obtains absolute knowledge of the characters in its visual window. Thus, there is no reason for the model to spend longer on one fixation than another, and the model only makes predictions for *where* saccades are targeted, and not *how long* fixations last.

Reichle and Laurent (2006) presented a rational model that overcame the limitations of Mr. Chips to produce predictions for both fixation durations and locations, focusing on the ways in which eye movement behavior is an adaptive response to the particular constraints of the task of reading. Given this focus, Reichle and Laurent used a very simple word identification function, for which the time required to identify a word was a function only of its length and the relative position of the eyes. In this paper, we present another rational model of eye movement control in reading that, like Reichle and Laurent, makes predictions for fixation durations and locations, but which focuses instead on the dynamics of word identification at the core of the task of reading. Specifically, our model identifies the words in a sentence by performing Bayesian inference combining noisy input from a realistic visual model with a language model that takes context into account.

3 Explaining between-word regressions

In this paper, we use our model to provide a novel explanation for between-word regressive saccades. In reading, about 10–15% of saccades are regressive – movements from right-to-left (or to previous lines). To understand how models such as *E-Z* Reader or SWIFT account for re-

gressive saccades to previous words, recall that the system identifies words in the sentence (generally) left to right, and that identification of a word in these models takes a certain amount of time and then is completed. In such a setup, why should the eyes ever move backwards? Three major answers have been put forward. One possibility given by *E-Z* Reader is as a response to overshoot; i.e., the eyes move backwards to a previous word because they accidentally landed further forward than intended due to motor error. Such an explanation could only account for small between-word regressions, of about the magnitude of motor error. The most recent version, *E-Z* Reader 10 (Reichle, Warren, & McConnell, 2009), has a new component that can produce longer between-word regressions. Specifically, the model includes a flag for postlexical integration failure, that – when triggered – will instruct the model to produce a between-word regression to the site of the failure. That is, between-word regressions in *E-Z* Reader 10 can arise because of postlexical processes external to the model's main task of word identification. A final explanation for between-word regressions, which arises as a result of normal processes of word identification, comes from the SWIFT model. In the SWIFT model, the reader can fail to identify a word but move past it and continue reading. In these cases, there is a chance that the eyes will at some point move back to this unidentified word to identify it. From the present perspective, however, it is unclear how it could be rational to move past an unidentified word and decide to revisit it only much later.

Here, we suggest a new explanation for between-word regressions that arises as a result of word identification processes (unlike that of *E-Z* Reader) and can be understood as rational (unlike that of SWIFT). Whereas in SWIFT and *E-Z* Reader, word recognition is a process that takes some amount of time and is then 'completed', some experimental evidence suggests that word recognition may be best thought of as a process that is never 'completed', as comprehenders appear to both maintain uncertainty about the identity of previous input and to update that uncertainty as more information is gained about the rest of the sentence (Connine, Blasko, & Hall, 1991; Levy, Bicknell, Slattery, & Rayner, 2009). Thus, it is possible that later parts of a sentence can cause a reader's confidence in the identity of the previous regions to fall. In these cases, a rational way to respond might be to make a between-word regressive saccade to get more visual information about the (now) low confidence previous region.

To illustrate this idea, consider the case of a language composed of just two strings, *AB* and *BA*, and assume that the eyes can only get noisy information about the identity of one character at a time. After obtaining a little information about the identity of the first character, the reader may be reasonably confident that its identity is *A* and move on to obtaining visual input about the second character. If the first noisy input about the second character also indicates that it is probably *A*, then the normative probability that the first character is *A* (and thus a rational reader's confidence in its identity) will fall. This simple example just illustrates the point that if a reader is combining noisy visual information with a language model, then confidence in previous regions will sometimes fall.

There are two ways that a rational agent might deal with this problem. The first option would be to reach a higher level of confidence in the identity of each word before moving on to the right, i.e., slowing down reading left-to-right to prevent having to make right-to-left regressions. The second option is to read left-to-right relatively more quickly, and then make occasional right-to-left regressions in the cases where probability in previous regions falls. In this paper, we present two simulations suggesting that when using a rational model to read natural language, the best strategies for coping with the problem of confidence about previous regions dropping – for any tradeoff between speed and accuracy – involve making between-word regressions. In the next section, we present the details of our model of reading and its implementation, and then we present our two simulations in the sections following.

4 Reading as Bayesian inference

At its core, the framework we are proposing is one of reading as Bayesian inference. Specifically, the model begins reading with a prior distribution over possible identities of a sentence given by its language model. On the basis of that distribution, the model decides whether or not to move its eyes (and if so where to move them to) and obtains noisy visual input about the sentence at the eyes' position. That noisy visual input then gives the likelihood term in a Bayesian belief update, where the

model's prior distribution over the identity of the sentence given the language model is updated to a posterior distribution taking into account both the language model and the visual input obtained thus far. On the basis of that new distribution, the model again selects an action and the cycle repeats.

This framework is unique among models of eye movement control in reading (except Mr. Chips) in having a fully explicit model of how visual input is used to discriminate word identity. This approach stands in sharp contrast to other models, which treat the time course of word identification as an exogenous function of other influencing factors (such as word length, frequency, and predictability). The hope in our approach is that the influence of these key factors on the eye movement record will fall out as a natural consequence of rational behavior itself. For example, it is well known that the higher the conditional probability of a word given preceding material, the more rapidly that word is read (Boston, Hale, Kliegl, Patil, & Vasishth, 2008; Demberg & Keller, 2008; Ehrlich & Rayner, 1981; Smith & Levy, 2008). *E-Z* Reader and SWIFT incorporate this finding by specifying a dependency on word predictability in the exogenous function determining word processing time. In our framework, in contrast, we would expect such an effect to emerge as a byproduct of Bayesian inference: words with high prior probability (conditional on preceding fixations) will require less visual input to be reliably identified.

An implemented model in this framework must formalize a number of pieces of the reading problem, including the possible actions available to the reader and their consequences, the nature of visual input, a means of combining visual input with prior expectations about sentence form and structure, and a control policy determining how the model will choose actions on the basis of its posterior distribution over the identities of the sentence. In the remainder of this section, we present these details of the formalization of the reading problem we used for the simulations reported in this paper: actions (4.1), visual input (4.2), formalization of the Bayesian inference problem (4.3), control policy (4.4), and finally, implementation of the model using weighted finite state automata (4.5).

4.1 Formal problem of reading: Actions

For our model, we assume a series of discrete timesteps, and on each time step, the model first obtains visual input around the current location of the eyes, and then chooses between three actions: (a) continuing to fixate the currently fixated position, (b) initiating a saccade to a new position, or (c) stopping reading of the sentence. If on the *i*th timestep, the model chooses option (a), the timestep advances to $i+1$ and another sample of visual input is obtained around the current position. If the model chooses option (c), the reading immediately ends. If a saccade is initiated (b), there is a lag of two timesteps, roughly representing the time required to plan and execute a saccade, during which the model again obtains visual input around the current position and then the eyes move – with some motor error – toward the intended target t_i, landing on position ℓ_i. On the next time step, visual input is obtained around ℓ_i and another decision is made. The motor error for saccades follows the form of random error used by all major models of eye movements in reading: the landing position ℓ_i is normally distributed around the intended target t_i with standard deviation given by a linear function of the intended distance[1]

$$\ell_i \sim \mathcal{N}\left(t_i, (\delta_0 + \delta_1 |t_i - \ell_{i-1}|)^2\right) \qquad (1)$$

for some linear coefficients δ_0 and δ_1. In the experiments reported in this paper, we follow the SWIFT model in using $\delta_0 = 0.87, \delta_1 = 0.084$.

4.2 Noisy visual input

As stated earlier, the role of noisy visual input in our model is as the likelihood term in a Bayesian inference about sentence form and identity. Therefore, if we denote the input obtained thus far from a sentence as $\mathcal{I}$, all the information pertinent to the reader's inferences can be encapsulated in the form $p(\mathcal{I}|w)$ for possible sentences w. We assume that the inputs deriving from each character position are conditionally independent given sentence identity, so that if w_j denotes letter j of the sentence and $\mathcal{I}(j)$ denotes the component of visual input associated with that letter, then we can decompose $p(\mathcal{I}|w)$ as $\prod_j p(\mathcal{I}(j)|w_j)$. For simplicity, we assume that each character is either a lowercase letter or a space. The visual input obtained from an individual fixation can thus be summarized as a vector of likelihoods $p(\mathcal{I}(j)|w_j)$, as shown in

[1] In the terminology of the literature, the model has only random motor error (variance), not systematic error (bias). Following Engbert and Krügel (2010), systematic error may arise from Bayesian estimation of the best saccade distance.

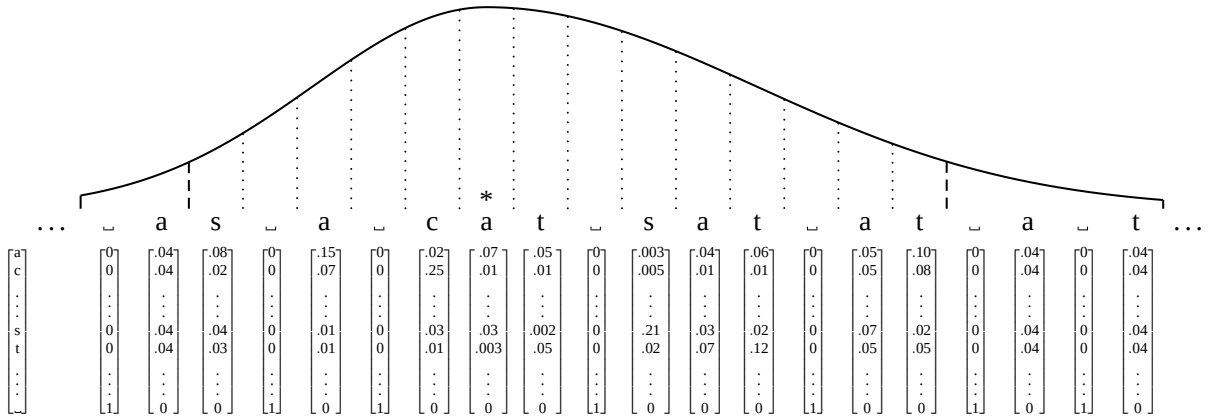


Figure 1: Peripheral and foveal visual input in the model. The asymmetric Gaussian curve indicates declining perceptual acuity centered around the fixation point (marked by $*$). The vector underneath each letter position denotes the likelihood $p(\mathcal{I}(j)|w_j)$ for each possible letter w_j, taken from a single input sample with $\Lambda = 1/\sqrt{3}$ (see vector at the left edge of the figure for key, and Section 4.2). In peripheral vision, the letter/whitespace distinction is veridical, but no information about letter identity is obtained. Note in this particular sample, input from the fixated character and the following one is rather inaccurate.

Figure 1. As in the real visual system, our visual acuity function decreases with retinal eccentricity; we follow the SWIFT model in assuming that the spatial distribution of visual processing rate follows an asymmetric Gaussian with $\sigma_L = 2.41, \sigma_R = 3.74$, which we discretize into processing rates for each character position. If ε denotes a character's eccentricity in characters from the center of fixation, then the proportion of the total processing rate at that eccentricity $\lambda(\varepsilon)$ is given by integrating the asymmetric Gaussian over a character width centered on that position,

$$\lambda(\varepsilon) = \int_{\varepsilon-.5}^{\varepsilon+.5} \frac{1}{Z} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx, \sigma = \begin{cases} \sigma_L, & x < 0 \\ \sigma_R, & x \geq 0 \end{cases}$$

where the normalization constant Z is given by

$$Z = \sqrt{\frac{\pi}{2}}(\sigma_L + \sigma_R).$$

From this distribution, we derive two types of visual input, *peripheral* input giving word boundary information and *foveal* input giving information about letter identity.

4.2.1 Peripheral visual input

In our model, any eccentricity with a processing rate proportion $\lambda(\varepsilon)$ at least 0.5% of the rate proportion for the centrally fixated character ($\varepsilon \in [-7, 12]$), yields peripheral visual input, defined as veridical word boundary information indicating whether each character is a letter or a space. This roughly corresponds to empirical estimates that humans obtain useful information in reading from about 19 characters, more from the right of fixation than the left (Rayner, 1998). Hence in Figure 1, for example, left-peripheral visual input can be represented as veridical knowledge of the initial whitespace (denoted ␣), and a uniform distribution over the 26 letters of English for the letter `a`.

4.2.2 Foveal visual input

In addition, for those eccentricities with a processing rate proportion $\lambda(\varepsilon)$ that is at least 1% of the total processing rate ($\varepsilon \in [-5, 8]$) the model receives foveal visual input, defined only for letters[2] to give noisy information about the letter's identity. This threshold of 1% roughly corresponds to estimates that readers get information useful for letter identification from about 4 characters to the left and 8 to the right of fixation (Rayner, 1998).

In our model, each letter is equally confusable with all others, following Norris (2006, 2009), but ignoring work on letter confusability (which could be added to future model revisions; Engel, Dougherty, & Jones, 1973; Geyer, 1977). Visual information about each character is obtained by sampling. Specifically, we represent each letter as a 26-dimensional vector, where a single element is 1 and the other 25 are zeros, and given this representation, foveal input for a letter is given as a sample from a 26-dimensional Gaussian with a

[2] For white space, the model is already certain of the identity because of peripheral input.

mean equal to the letter's true identity and a diagonal covariance matrix $\Sigma(\varepsilon) = \lambda(\varepsilon)^{-1/2}I$. It is relatively straightforward to show that under these conditions, if we take the processing rate to be the expected change in log-odds of the true letter identity relative to any other that a single sample brings about, then the rate equals $\lambda(\varepsilon)$. We scale the overall processing rate by multiplying each rate by Λ. For the experiments in this paper, we set $\Lambda = 4$. For each fixation, we sample independently from the appropriate distribution for each character position and then compute the likelihood given each possible letter, as illustrated in the non-peripheral region of Figure 1.

4.3 Inference about sentence identity

Given the visual input and a language model, inferences about the identity of the sentence w can be made by standard Bayesian inference, where the prior is given by the language model and the likelihood is a function of the total visual input obtained from the first to the ith timestep $\mathcal{I}_1^i$,

$$p(w|\mathcal{I}_1^i) = \frac{p(w)p(\mathcal{I}_1^i|w)}{\sum_{w'}(w')p(\mathcal{I}_1^i|w')}. \quad (2)$$

If we let $\mathcal{I}(j)$ denote the input received about character position j and let w_j denote the jth character in sentence identity w, then the likelihood can be broken down by character position as

$$p(\mathcal{I}_1^i|w) = \prod_{j=1}^{n} p(\mathcal{I}_1^i(j)|w_j)$$

where n is the final character about which there is any visual input. Similarly, we can decompose this into the product of the likelihoods of each sample

$$p(\mathcal{I}_1^i|w) = \prod_{j=1}^{n}\prod_{t=1}^{i} p(\mathcal{I}_t(j)|w_j). \quad (3)$$

If the eccentricity of the jth character on the tth timestep ε_t^j is outside of foveal input or the character is a space, the inner term is 0 or 1. If the sample was from a letter in foveal input $\varepsilon_t^j \in [-5, 8]$, it is the probability of sampling $\mathcal{I}_t(j)$ from the multivariate Gaussian $\mathcal{N}(w_j, \Lambda\Sigma(\varepsilon_t^j))$.

4.4 Control policy

The model uses a simple policy to decide between actions based on the marginal probability m of the most likely character c in position j,

$$m(j) = \max_c p(w_n = c|\mathcal{I}_1^i) = \max_c \sum_{w':w'_n=c} p(w'|\mathcal{I}_1^i). \quad (4)$$

Intuitively, a high value of m means that the model is relatively confident about the character's identity, and a low value that it is relatively uncertain.

Given the values of this statistic, our model decides between four possible actions, as illustrated in Figure 2. If the value of this statistic for the current position of the eyes $m(\ell_i)$ is less than a parameter α, the model chooses to continue fixating the current position (2a). Otherwise, if the value of $m(j)$ is less than β for some leftward position $j < \ell_i$, the model initiates a saccade to the closest such position (2b). If $m(j) \geq \beta$ for all $j < \ell_i$, then the model initiates a saccade to n characters past the closest position to the right $j > \ell_i$ for which $m(j) < \alpha$ (2c).[3] Finally, if no such positions exist to the right, the model stops reading the sentence (2d). Intuitively, then, the model reads by making a rightward sweep to bring its confidence in each character up to α, but pauses to move left if confidence in a previous character falls below β.

(a) $m = [.6, .7, \mathbf{.6}, .4, .3, .6]$: Keep fixating (3)
(b) $m = [.6, .4, \mathbf{.9}, .4, .3, .6]$: Move back (to 2)
(c) $m = [.6, .7, \mathbf{.9}, .4, .3, .6]$: Move forward (to 6)
(d) $m = [.6, .7, \mathbf{.9}, .8, .7, .7]$: Stop reading

Figure 2: Values of m for a 6 character sentence under which a model fixating position 3 would take each of its four actions, if $\alpha = .7$ and $\beta = .5$.

4.5 Implementation with wFSAs

This model can be efficiently and simply implemented using weighted finite-state automata (wFSAs; Mohri, 1997) as follows: First, we begin with a wFSA representation of the language model, where each arc emits a single character (or is an epsilon-transition emitting nothing). To perform belief update given a new visual input, we create a new wFSA to represent the likelihood of each character from the sample. Specifically, this wFSA has only a single chain of states, where, e.g., the first and second state in the chain are connected by 27 (or fewer) arcs, which emit each of

[3] The role of n is to ensure that the model does not center its visual field on the first uncertain character. We did not attempt to optimize this parameter, but fixed n at 2.

the possible characters for w_1 along with their respective likelihoods given the visual input (as in the inner term of Equation 3). Next, these two wFSAs may simply be composed and then normalized, which completes the belief update, resulting in a new wFSA giving the posterior distribution over sentences. To calculate the statistic m, while it is possible to calculate it in closed form from such a wFSA relatively straightforwardly, for efficiency we use Monte Carlo estimation based on samples from the wFSA.

5 Simulation 1

With the description of our model in place, we next proceed to describe the first simulation in which we used the model to test the hypothesis that making regressions is a rational way to cope with confidence in previous regions falling. Because there is in general no single rational tradeoff between speed and accuracy, our hypothesis is that, for any given level of speed and accuracy achieved by a non-regressive policy, there is a faster and more accurate policy that makes a faster left-to-right pass but occasionally does make regressions. In the terms of our model's policy parameters α and β described above, non-regressive policies are exactly those with $\beta = 0$, and a policy that is faster on the left-to-right pass but does make regressions is one with a lower value of α but a non-zero β. Thus, we tested the performance of our model on the reading of a corpus of text typical of that used in reading experiments at a range of reasonable non-regressive policies, as well as a set of regressive policies with lower α and positive β. Our prediction is that the former set will be strictly dominated in terms of both speed and accuracy by the latter.

5.1 Methods

5.1.1 Policy parameters

We test 4 non-regressive policies (i.e., those with $\beta = 0$) with values of $\alpha \in \{.90, .95, .97, .99\}$, and in addition, test regressive policies with a lower range of $\alpha \in \{.85, .90, .95, .97\}$ and $\beta \in \{.4, .7\}$.[4]

5.1.2 Language model

Our reader's language model was an unsmoothed bigram model created using a vocabulary set consisting of the 500 most frequent words in the British National Corpus (BNC) as well as all the words in our test corpus. From this vocabulary, we constructed a bigram model using the counts from every bigram in the BNC for which both words were in vocabulary (about 222,000 bigrams).

5.1.3 wFSA implementation

We implemented our model with wFSAs using the OpenFST library (Allauzen, Riley, Schalkwyk, Skut, & Mohri, 2007). Specifically, we constructed the model's initial belief state (i.e., the distribution over sentences given by its language model) by directly translating the bigram model into a wFSA in the log semiring. We then composed this wFSA with a weighted finite-state transducer (wFST) breaking words down into characters. This was done in order to facilitate simple composition with the visual likelihood wFSA defined over characters. In the Monte Carlo estimation of m, we used 5000 samples from the wFSA. Finally, to speed performance, we bounded the wFSA to have exactly the number of characters present in the actual sentence and then renormalized.

5.1.4 Test corpus

We tested our model's performance by simulating reading of the Schilling corpus (Schilling, Rayner, & Chumbley, 1998). To ensure that our results did not depend on smoothing, we only tested the model on sentences in which every bigram occurred in the BNC. Unfortunately, only 8 of the 48 sentences in the corpus met this criterion. Thus, we made single-word changes to 25 more of the sentences (mostly changing proper names and rare nouns) to produce a total of 33 sentences to read, for which every bigram did occur in the BNC.

5.2 Results and discussion

For each policy we tested, we measured the average number of timesteps it took to read the sentences, as well as the average (natural) log probability of the correct sentence identity under the model's beliefs after reading ended 'Accuracy'. The results are plotted in Figure 3. As shown in the graph, for each non-regressive policy (the circles), there is a regressive policy that outperforms it, both in terms of average number of timesteps taken to read (further to the left) and the average log probability of the sentence identity (higher). Thus, for a range of policies, these results suggest

[4] We tested all combinations of these values of α and β except for $[\alpha, \beta] = [.97, .4]$, because we did not believe that a value of β so low in relation to α would be very different from a non-regressive policy.

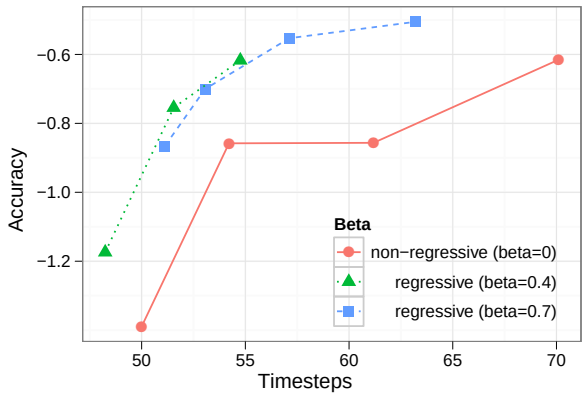


Figure 3: Mean number of timesteps taken to read a sentence and (natural) log probability of the true identity of the sentence 'Accuracy' for a range of values of α and β. Values of α are not labeled, but increase with the number of timesteps for a constant value of β. For each non-regressive policy ($\beta = 0$), there is a policy with a lower α and higher β that achieves better accuracy in less time.

that making regressions when confidence about previous regions falls is a rational reader strategy, in that it appears to lead to better performance, both in terms of speed and accuracy.

6 Simulation 2

In Simulation 2, we perform a more direct test of the idea that making regressions is a rational response to the problem of confidence falling about previous regions using optimization techniques. Specifically, we search for optimal policy parameter values (α, β) for three different measures of performance, each representing a different tradeoff between the importance of accuracy and speed.

6.1 Methods

6.1.1 Performance measures

We examine performance measures interpolating between speed and accuracy of the form

$$L(1-\gamma) - T\gamma \quad (5)$$

where L is the log probability of the true identity of the sentence under the model's beliefs at the end of reading, and T is the total number of timesteps before the model decided to stop reading. Thus, each different performance measure is determined by the weighting for time γ. We test three values of $\gamma \in \{.025, .1, .4\}$. The first of these weights accuracy highly, while the final one weights 1 timestep almost as much as 1 unit of log probability.

6.1.2 Optimization of policy parameters

Searching directly for optimal values of α and β for our stochastic reading model is difficult because each evaluation of the model with a particular set of parameters produces a different result. We use the PEGASUS method (Ng & Jordan, 2000) to transform this stochastic optimization problem into a deterministic one on which we can use standard optimization algorithms.[5] Then, we evaluate the model's performance at each value of α and β by reading the full test corpus and averaging performance. We then simply use coordinate ascent (in logit space) to find the optimal values of α and β for each performance measure.

6.1.3 Language model

The language model used in this simulation begins with the same vocabulary set as in Sim. 1, i.e., the 500 most frequent words in the BNC and every word that occurs in our test corpus. Because the search algorithm demands that we evaluate the performance of our model at a number of parameter values, however, it is too slow to optimize α and β using the full language model that we used for Sim. 1. Instead, we begin with the same set of bigrams used in Sim. 1 – i.e., those that contain two in-vocabulary words – and trim this set by removing rare bigrams that occur less than 200 times in the BNC (except that we do not trim any bigrams that occur in our test corpus). This reduces our set of bigrams to about 19,000.

6.1.4 wFSA implementation

The implementation was the same as in Sim. 1.

6.1.5 Test corpus

The test corpus was the same as in Sim. 1.

6.2 Results and discussion

The optimal values of α and β for each $\gamma \in \{.025, .1, .4\}$ are given in Table 1 along with the mean values for L and T found at those parameter values. As the table shows, the optimization procedure successfully found values of α and β, which go up (slower reading) as γ goes down (valuing accuracy more than time). In addition, we see that the average results of reading at these parameter values are also as we would expect, with T and L going up as γ goes down. As predicted, the optimal

[5] Specifically, this involves fixing the random number generator for each run to produce the same values, resulting in minimizing the variance in performance across evaluations.

γ	α	β	Timesteps	Log probability
.025	.90	.99	41.2	-0.02
.1	.36	.80	25.8	-0.90
.4	.18	.38	16.4	-4.59

Table 1: Optimal values of α and β found for each performance measure γ tested and mean performance at those values, measured in timesteps T and (natural) log probability L.

values of β found are non-zero across the range of policies, which include policies that value speed over accuracy much more than in Sim. 1. This provides more evidence that whatever the particular performance measure used, policies making regressive saccades when confidence in previous regions falls perform better than those that do not.

There is one interesting difference between the results of this simulation and those of Sim. 1, which is that here, the optimal policies all have a value of $\beta > \alpha$. That may at first seem surprising, since the model's policy is to fixate a region until its confidence becomes greater than α and then return if it falls below β. It would seem, then, that the only reasonable values of β are those that are strictly below α. In fact, this is not the case because of the two time step delay between the decision to move the eyes and the execution of that saccade. Because of this delay, the model's confidence when it leaves a region (relevant to β) will generally be higher than when it decided to leave (determined by α). In Simulation 2, because of the smaller grammar that was used, the model's confidence in a region's identity rises more quickly and this difference is exaggerated.

7 Conclusion

In this paper, we presented a model that performs Bayesian inference on the identity of a sentence, combining a language model with noisy information about letter identities from a realistic visual input model. On the basis of these inferences, it uses a simple policy to determine how long to continue fixating the current position and where to fixate next, on the basis of information about where the model is uncertain about the sentence's identity. As such, it constitutes a rational model of eye movement control in reading, extending the insights from previous results about rationality in language comprehension.

The results of two simulations using this model support a novel explanation for between-word regressive saccades in reading: that they are used to gather visual input about previous regions when confidence about them falls. Simulation 1 showed that a range of policies making regressions in these cases outperforms a range of non-regressive policies. In Simulation 2, we directly searched for optimal values for the policy parameters for three different performance measures, representing different speed-accuracy trade-offs, and found that the optimal policies in each case make substantial use of between-word regressions when confidence in previous regions falls. In addition to supporting a novel motivation for between-word regressions, these simulations demonstrate the possibility for testing a range of questions that were impossible with previous models of reading related to the goals of a reader, such as how should reading behavior change as accuracy is valued more.

There are a number of obvious ways for the model to move forward. One natural next step is to make the model more realistic by using letter confusability matrices. In addition, the link to previous work in sentence processing can be made tighter by incorporating syntax-based language models. It also remains to compare this model's predictions to human data more broadly on standard benchmark measures for models of reading. The most important future development, however, will be moving toward richer policy families, which enable more intelligent decisions about eye movement control, based not just on simple confidence statistics calculated independently for each character position, but rather which utilize the rich structure of the model's posterior beliefs about the sentence identity (and of language itself) to make more informed decisions about the best time to move the eyes and the best location to direct them next.

Acknowledgments

The authors thank Jeff Elman, Tom Griffiths, Andy Kehler, Keith Rayner, and Angela Yu for useful discussion about this work. This work benefited from feedback from the audiences at the 2010 LSA and CUNY conferences. The research was partially supported by NIH Training Grant T32-DC000041 from the Center for Research in Language at UC San Diego to K.B., by a research grant from the UC San Diego Academic Senate to R.L., and by NSF grant 0953870 to R.L.

References

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., & Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)* (Vol. 4783, p. 11-23). Springer.

Bicknell, K., & Levy, R. (2010). Rational eye movements in reading combining uncertainty about previous words with contextual probability. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society.* Austin, TX: Cognitive Science Society.

Boston, M. F., Hale, J. T., Kliegl, R., Patil, U., & Vasishth, S. (2008). Parsing costs as predictors of reading difficulty: An evaluation using the potsdam sentence corpus. *Journal of Eye Movement Research*, *2*(1), 1–12.

Connine, C. M., Blasko, D. G., & Hall, M. (1991). Effects of subsequent sentence context in auditory word recognition: Temporal and linguistic constraints. *Journal of Memory and Language*, *30*, 234–250.

Demberg, V., & Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, *109*, 193–210.

Ehrlich, S. F., & Rayner, K. (1981). Contextual effects on word perception and eye movements during reading. *Journal of Verbal Learning and Verbal Behavior*, *20*, 641–655.

Engbert, R., & Krügel, A. (2010). Readers use Bayesian estimation for eye movement control. *Psychological Science*, *21*, 366–371.

Engbert, R., Longtin, A., & Kliegl, R. (2002). A dynamical model of saccade generation in reading based on spatially distributed lexical processing. *Vision Research*, *42*, 621–636.

Engbert, R., Nuthmann, A., Richter, E. M., & Kliegl, R. (2005). SWIFT: A dynamical model of saccade generation during reading. *Psychological Review*, *112*, 777–813.

Engel, G. R., Dougherty, W. G., & Jones, B. G. (1973). Correlation and letter recognition. *Canadian Journal of Psychology*, *27*, 317–326.

Genzel, D., & Charniak, E. (2002, July). Entropy rate constancy in text. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 199–206). Philadelphia: Association for Computational Linguistics.

Genzel, D., & Charniak, E. (2003). Variation of entropy and parse trees of sentences as a function of the sentence number. In M. Collins & M. Steedman (Eds.), *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing* (pp. 65–72). Sapporo, Japan: Association for Computational Linguistics.

Geyer, L. H. (1977). Recognition and confusion of the lowercase alphabet. *Perception & Psychophysics*, *22*, 487–490.

Hale, J. (2001). A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics* (Vol. 2, pp. 159–166). New Brunswick, NJ: Association for Computational Linguistics.

Jaeger, T. F. (2010). Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology*. doi:10.1016/j.cogpsych.2010.02.002.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, *20*, 137–194.

Keller, F. (2004). The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In D. Lin & D. Wu (Eds.), *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing* (pp. 317–324). Barcelona, Spain: Association for Computational Linguistics.

Legge, G. E., Hooven, T. A., Klitz, T. S., Mansfield, J. S., & Tjan, B. S. (2002). Mr. Chips 2002: new insights from an ideal-observer model of reading. *Vision Research*, *42*, 2219–2234.

Legge, G. E., Klitz, T. S., & Tjan, B. S. (1997). Mr. Chips: an Ideal-Observer model of reading. *Psychological Review*, *104*, 524–553.

Levy, R. (2008). A noisy-channel model of rational human sentence comprehension under uncertain input. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* (pp. 234–243). Honolulu, Hawaii: Association for Computational Linguistics.

Levy, R., Bicknell, K., Slattery, T., & Rayner, K. (2009). Eye movement evidence that readers maintain and act on uncertainty about past linguistic input. *Proceedings of the National Academy of Sciences*, *106*, 21086–21090.

Levy, R., & Jaeger, T. F. (2007). Speakers optimize information density through syntactic reduction. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19* (pp. 849–856). Cambridge, MA: MIT Press.

Levy, R., Reali, F., & Griffiths, T. L. (2009). Modeling the effects of memory on human online sentence processing with particle filters. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems 21* (pp. 937–944).

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, *23*, 269–311.

Narayanan, S., & Jurafsky, D. (2001). A Bayesian model predicts human parse preference and reading time in sentence processing. In T. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14* (pp. 59–65). Cambridge, MA: MIT Press.

Ng, A. Y., & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference* (pp. 406–415).

Norris, D. (2006). The Bayesian reader: Explaining word recognition as an optimal Bayesian decision process. *Psychological Review*, *113*, 327–357.

Norris, D. (2009). Putting it all together: A unified account of word recognition and reaction-time distributions. *Psychological Review*, *116*, 207–219.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, *124*, 372–422.

Reichle, E. D., & Laurent, P. A. (2006). Using reinforcement learning to understand the emergence of "intelligent" eye-movement behavior during reading. *Psychological Review*, *113*, 390–408.

Reichle, E. D., Pollatsek, A., Fisher, D. L., & Rayner, K. (1998). Toward a model of eye movement control in reading. *Psychological Review*, *105*, 125–157.

Reichle, E. D., Pollatsek, A., & Rayner, K. (2006). E-Z Reader: A cognitive-control, serial-attention model of eye-movement behavior during reading. *Cognitive Systems Research*, *7*, 4–22.

Reichle, E. D., Warren, T., & McConnell, K. (2009). Using E-Z Reader to model the effects of higher level language processing on eye movements during reading. *Psychonomic Bulletin & Review*, *16*, 1–21.

Schilling, H. E. H., Rayner, K., & Chumbley, J. I. (1998). Comparing naming, lexical decision, and eye fixation times: Word frequency effects and individual differences. *Memory & Cognition*, *26*, 1270–1281.

Smith, N. J., & Levy, R. (2008). Optimal processing times in reading: a formal model and empirical investigation. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 595–600). Austin, TX: Cognitive Science Society.

Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science*, *268*, 1632–1634.

The Influence of Discourse on Syntax
A Psycholinguistic Model of Sentence Processing

Amit Dubey

Abstract

Probabilistic models of sentence comprehension are increasingly relevant to questions concerning human language processing. However, such models are often limited to syntactic factors. This paper introduces a novel sentence processing model that consists of a parser augmented with a probabilistic logic-based model of coreference resolution, which allows us to simulate how context interacts with syntax in a reading task. Our simulations show that a Weakly Interactive cognitive architecture can explain data which had been provided as evidence for the Strongly Interactive hypothesis.

1 Introduction

Probabilistic grammars have been found to be useful for investigating the architecture of the human sentence processing mechanism (Jurafsky, 1996; Crocker and Brants, 2000; Hale, 2003; Boston et al., 2008; Levy, 2008; Demberg and Keller, 2009). For example, probabilistic models shed light on so-called locality effects: contrast the non-probabilistic hypothesis that dependants which are far away from their head *always* cause processing difficulty for readers due to the cost of storing the intervening material in memory (Gibson, 1998), compared to the probabilistic prediction that there are cases when faraway dependants facilitate processing, because readers have more time to predict the head (Levy, 2008). Using a computational model to address fundamental questions about sentence comprehension motivates the work in this paper.

So far, probabilistic models of sentence processing have been largely limited to syntactic factors. This is unfortunate because many outstanding questions in psycholinguistics concern interactions between different levels of processing. This paper addresses this gap by building a computational model which simulates the influence of discourse on syntax.

Going beyond the confines of syntax alone is a sufficiently important problem that it has attracted attention from other authors. In the literature on probabilistic modeling, though, the bulk of this work is focused on lexical semantics (e.g. Padó et al., 2006; Narayanan and Jurafsky, 1998) or only considers syntactic decisions in the preceeding text (e.g. Dubey et al., 2009; Levy and Jaeger, 2007). This is the first model we know of which introduces a broad-coverage sentence processing model which takes the effect of coreference and discourse into account.

A major question concerning discourse-syntax interactions involves the strength of communication between discourse and syntactic information. The **Weakly Interactive** (Altmann and Steedman, 1988) hypothesis states that a discourse context can reactively prune syntactic choices that have been proposed by the parser, whereas the **Strongly Interactive** hypothesis posits that context can proactively suggest choices to the syntactic processor.

Support for Weak Interaction comes from experiments in which there are temporary ambiguities, or *garden paths*, which cause processing difficulty. The general finding is that supportive contexts can reduce the effect of the garden path. However, Grodner et al. (2005) found that supportive contexts even facilitate the processing of *unambiguous* sentences. As there are no incorrect analyses to prune in unambiguous structures, the authors claimed their results were not consistent with the Weakly Interactive hypothesis, and suggested that their results were best explained by a Strongly Interactive processor.

The model we present here implements the Weakly Interactive hypothesis, but we will show that it can nonetheless successfully simulate the results of Grodner et al. (2005). There are three main parts of the model: a syntactic processor, a coreference resolution system, and a simple pragmatics processor which computes certain limited forms of discourse coherence. Following Hale (2001) and Levy (2008), among others, the syntactic processor uses an incremental probabilistic Earley parser to compute a metric which correlates with increased reading difficulty. The coreference resolution system is implemented

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1179–1188,
Uppsala, Sweden, 11-16 July 2010.

in a *probabilistic logic* known as Markov Logic (Richardson and Domingos, 2006). Finally, the pragmatics processing system contains a small set of probabilistic constraints which convey some intuitive facts about discourse processing. The three components form a pipeline, where each part is probabilistically dependent on the previous one. This allows us to combine all three into a single probability for each reading of an input sentence.

The rest of the paper is structured as follows. In Section 2, we discuss the details two experiments showing support of the Weakly and Strongly Interactive hypotheses: we discuss Grodner et al.'s result on unambiguous syntactic structures and we present a new experiment on involving a garden path which was designed to be similar to the Grodner et al. experiment. Section 3 introduces technical details of model, and Section 4 shows the predictions of the model on the experiments discussed in Section 2. Finally, we discuss the theoretical consequences of these predictions in Section 5.

2 Cognitive Experiments

2.1 Discourse and Ambiguity Resolution

There is a fairly large literature on garden path experiments involving context (Crain and Steedman, 1985; Mitchell et al., 1992, ibid). The experiments by Altmann and Steedman (1988) involved PP attachment ambiguity. Other authors (e.g. Spivey and Tanenhaus, 1998) have used reduced relative clause attachment ambiguity. In order to be more consistent with the design of the experiment in Section 2.2, however, we performed our own reading-time experiment which partially replicated previous results.[1]

The experimental items all had a target sentence containing a relative clause, and one of two possible context sentences, one of which supports the relative clause reading and the other which does not.

The context sentence was one of:

(1) a. There were two postmen, one of whom was injured and carried by paramedics, and another who was unhurt.
b. Although there was a medical emergency at the post office earlier today, regular mail delivery was unaffected.

[1]This experiment was previously reported by Dubey et al. (2010).

The target sentences, which were drawn from the experiment of McRae et al. (1998), were either the reduced or unreduced sentences similar to:

(2) The postman *who was* carried by the paramedics was having trouble breathing.

The reduced version of the sentence is produced by removing the words *who was*. We measured reading times in the underlined region, which is the first point at which there is evidence for the relative clause interpretation. The key evidence is given by the word 'by', but the previous word is included as readers often do not fixate on short function words, but rather process them while overtly fixating on the previous word (Rayner, 1998).

The relative clauses in the target sentence act as *restrictive* relative clauses, selecting one referent from a larger set. The target sentences are therefore more coherent in a context where a restricted set and a contrast set are easily available, than one in which these sets are absent. This makes the context in Example (1-a) supportive of a reduced relative reading, and the context in Example (1-b) unsupportive of a reduced relative clause. Other experiments, for instance Spivey and Tanenhaus (1998), used an unsupportive context where only one postman was mentioned. Our experiments used a neutral context, where no postmen are mentioned, to be more similar to the Grodner et al. experiment, as described below.

Overall, there were 28 items, and 28 participants read these sentences using an EyeLink II eyetracker. Each participant read items one at a time, with fillers between subsequent items so as to obfuscate the nature of the experiment.

Results An ANOVA revealed that all conditions with a supportive context were read faster than one with a neutral context (i.e. a main effect of context), and all conditions with unambiguous syntax were read faster than those with a garden path (i.e. a main effect of ambiguity). Finally, there was a statisically significant interaction between syntax and discourse whereby context decreases reading times much more when a garden path is present compared to an unambiguous structure. In other words, a supportive context helped reduce the effect of a garden path. This is the prediction made by both the Weakly Interactive and Strongly Interactive hypothesis. The pattern of results are shown in Figure 2a in Section 4, where they are directly compared to the model results.

2.2 Discourse and Unambiguous Syntax

As mentioned in the Introduction, Grodner et al. (2005) proposed an experiment with a supportive or unsupportive discourse followed by an unambiguous target sentence. In their experiment, the target sentence was one of the following:

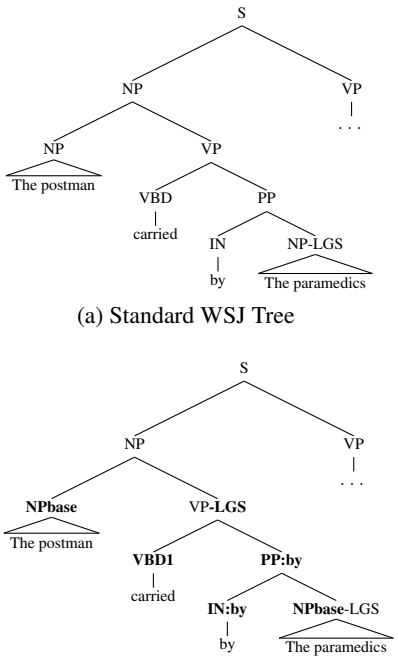
(3) a. The director that the critics praised at a banquet announced that he was retiring to make room for young talent in the industry.
b. The director, who the critics praised at a banquet, announced that he was retiring to make room for young talent in the industry.

They also manipulated the context, which was either supportive of the target, or a null context. The two supportive contexts are:

(4) a. A group of film critics praised a director at a banquet and another director at a film premiere.
b. A group of film critics praised a director and a producer for lifetime achievement.

The target sentence in (3-a) is a restrictive relative clause, as in the garden path experiments. However, the sentence in (3-b) is a non-restrictive relative clause, which does not assume the presence of a constrast set. Therefore, the context (4-a) is only used with the restrictive relative clause, and the context (4-b), where only one director is mentioned, is used as the context for the non-restrictive relative clause. In the conditions with a null context, the target sentence was not preceded by any contextual sentence.

Results Grodner et al. measured residual reading times, i.e. reading times compared to a baseline in the embedded subject NP ('the critics'). They found that the supportive contexts decreased reading time, and that this effect was stronger for restrictive relatives compared to non-restricted relatives. As there was no garden path, and hence no incorrect structure for the discourse processor to prune, the authors conclude that this must be evidence for the Strongly Interactive hypothesis. Unlike the garden path experiment above, these results do not appear to be consistent with a Weakly Interactive model. We plot their results in Figure 3a in Section 4, where they are directly compared to the model results. Because these results are computed as regressions against a baseline, a reading time of 0ms indicates average difficulty, with negative numbers showing some facilitation has occured, and positive number indicating reading difficulty.



(a) Standard WSJ Tree

(b) Minimally Modified Tree

Figure 1: A schematic representation of the smallest set of grammar transformations which we found were required to accurately parse the experimental items.

3 Model

The model comprises three parts: a parser, a coreference resolution system, and a pragmatics subsystem. Let us look at each individually.

3.1 Parser

The parser is an incremental unlexicalized probabilistic Earley parser, which is capable of computing prefix probabilities. A PCFG parser outputs the generative probability $P_{parser}(w, t)$, where w is the text and t is a parse tree. A probabilistic Earley parser can retrieve all possible derivations at word i (Stolcke, 1995), allowing us to compute the probability $P(w_i \dots w_0) = \sum_t P_{parser}(w_i \dots w_0, t)$.

Using the prefix probability, we can compute the word-by-word Surprisal (Hale, 2001), by taking the log ratio of the previous word's prefix probability against this word's prefix probability:

$$\log \left(\frac{P(w_{i-1} \dots w_0)}{P(w_i \dots w_0)} \right) \quad (1)$$

Higher Surprisal scores are interpreted as

being correlated with more reading difficulty, and likewise lower scores with greater reading ease. For most of the remainder of the paper we will simply refer to the prefix probability at word i as $P(w)$. While the prefix probability as presented here is suitable for syntax-based computations, a main technical contribution of our model, detailed in Sections 3.2 and 3.3 below, is that we include non-syntactic probabilities in the computation of Surprisal.

As per Hale's original suggestion, our parser can compute Surprisal using an exhaustive search, which entails summing over each licensed derivation. This can be done efficiently using the packed representation of an Earley chart. However, as the coreference processor takes trees as input, we must therefore unpack parses before resolving referential ambiguity. Given the ambiguity of our grammar, this is not tractable. Therefore, we only consider an n-best list when computing Surprisal. As other authors have found that a relatively small set of analyses can give meaningful predictions (Brants and Crocker, 2000; Boston et al., 2008), we set $n = 10$.

The parser is trained on the Wall Street Journal (WSJ) section of the Penn treebank. Unfortunately, the standard WSJ grammar is not able to give correct incremental parses to our experimental items. We found we could resolve this problem by using four simple transformations, which are shown in Figure 1: (i) adding valency information to verb POS tags (e.g. VBD1 represents a transitive verb); (ii) we lexicalize 'by' prepositions; (iii) VPs containing a logical subject (i.e. the agent), get the -LGS label; (iv) non-recursive NPs are renamed NPbase (the coreference system treats each NPbase as a markable).

3.2 Discourse Processor

The primary function of the discourse processing module is to perform coreference resolution for each mention in an incrementally processed text. Because each mention in a coreference chains is transitive, we cannot use a simple classifier, as they cannot enforce global transitivity constraints. Therefore, this system is implemented in Markov Logic (Richardson and Domingos, 2006), a probabilistic logic, which does allow us to include such constraints.

Markov Logic attempts to combine logic with probabilities by using a Markov random field where logical formulas are features. The Markov Logic Network (MLN) we used for our system uses similar predicates as the MLN-based coreference resolution system of Huang et al. (2009).[2] Our MLN uses the predicates listed in Table 1. Two of these predicates, $Coref$ and $First$, are the output of the MLN – they provide a labelling of coreference mentions into entity classes. Note that, unlike Huang et al., we assume an ordering on x and y if $Coref(x, y)$ is true: y must occur earlier in the document than x. The remaining predicates in Table 1 are a subset of features used by other coreference resolution systems (cf. Soon et al., 2001). The predicates we use involve matching strings (checking if two mentions share a head word or if they are exactly the same string), matching argreement features (if the gender, number or person of pairs of NPs are the same; especially important for pronouns), the distance between mentions, and if mentions have the same entity type (i.e. do they refer to a person, organization, etc.) As our main focus is not to produce a state-of-the-art coreference system, we do not include predicates which are irrevelant for our simulations even if they have been shown to be effective for coreference resolution. For example, we do not have predicates if two mentions are in an apposition relationship, or if two mentions are synonyms for each other.

Expression	Meaning
$Coref(x, y)$	x is coreferent with y.
$First(x)$	x is a first mention.
$Order(x, y)$	x occurs before y.
$SameHead(x, y)$	Do x and y share the same syntactic head?
$ExactMatch(x, y)$	x and y are same string.
$SameNumber(x, y)$	x and y match in number.
$SameGender(x, y)$	x and y match in gender.
$SamePerson(x, y)$	x and y match in person.
$Distance(x, y, d)$	The distance between x and y, in sentences.
$Pronoun(x)$	x is a pronoun.
$EntityType(x, e)$	x has entity type e (person, organization, etc.)

Table 1: Predicates used in the Markov Logic Network

Table 2 lists the actual logical formulae which are used as features in the MLN. It should be

[2] As we are not interested in unsupervised inference, the system of Poon and Domingos (2008) was unsuitable for our needs.

Description	Rule
Transitivity	$Coref(x,z) \wedge Coref(y,z) \wedge Order(x,y) \Rightarrow Coref(x,y)$
	$Coref(x,y) \wedge Coref(y,z) \Rightarrow Coref(x,z)$
	$Coref(x,y) \wedge Coref(x,z) \wedge Order(y,z) \Rightarrow Coref(y,z)$
First Mentions	$Coref(x,y) \Rightarrow \neg First(x)$
	$First(x) \Rightarrow \neg Coref(x,y)$
String Match	$ExactMatch(x,y) \Rightarrow Coref(x,y)$
	$SameHead(x,y) \Rightarrow Coref(x,y)$
Pronoun	$Pronoun(x) \wedge Pronoun(y) \wedge SameGender(x,y) \Rightarrow Coref(x,y)$
	$Pronoun(x) \wedge Pronoun(y) \wedge SameNumber(x,y) \Rightarrow Coref(x,y)$
	$Pronoun(x) \wedge Pronoun(y) \wedge SamePerson(x,y) \Rightarrow Coref(x,y)$
Other	$EntityType(x,e) \wedge EntityType(y,e) \Rightarrow Coref(x,y)$
	$Distance(x,y,+d) \Rightarrow Coref(x,y)$

Table 2: Rules used in the Markov Logic Network

noted that, because we are assuming an order on the arguments of $Coref(x,y)$, we need three formulae to capture transivity relationships. To test that the coreference resolution system was producing meaningful results, we evaluated our system on the test section of the ACE-2 dataset. Using b^3 scoring (Bagga and Baldwin, 1998), which computes the overlap of a proposed set with the gold set, the system achieves an F-score of 65.4%. While our results are not state-of-the-art, they are reasonable considering the brevity of our feature list.

The discourse model is run iteratively at each word. This allows us to find a globally best assignment at each word, which can be reanalyzed at a later point in time. It assumes there is a mention for each base NP outputted by the parser, and for all ordered pairs of mentions x, y, it outputs all the 'observed' predicates (i.e. everything but $First$ and $Coref$), and feeds them to the Markov Logic system. At each step, we compute both the maximum a posteriori (MAP) assignment of coreference relationships as well as the probability that each individual coreference assignment is true. Taken together, they allow us to calculate, for a coreference assignment c, $P_{coref}(c|w,t)$ where w is the text input (of the entire document until this point), and t is the parse of each tree in the document up to and including the current incremental parse. As we have previously calculated $P_{parser}(w,t)$, it is then possible to compute the joint probability $P(c,w,t)$ at each word, and therefore the prefix probability $P(w)$ due to syntax and coreference. Overall, we have:

$$\begin{aligned} P(w) &= \sum_c \sum_t P(c,w,t) \\ &= \sum_c \sum_t P_{coref}(c|w,t) P_{parser}(w,t) \end{aligned}$$

Note that we only consider *one* possible assignment of NPs to coreference entities per parse, as we only retrieve the probabilities of the MAP solution.

3.3 Pragmatics Processor

The effect of context in the experiments described in Section 2 cannot be fully explained using a coreference resolution system alone. In the case of restrictive relative clauses, the referential 'mismatch' in the unsupported conditions is caused by an expectation elicited by a restrictive relative clause which is inconsistent with the previous discourse when there is no salient restricted subset of a larger set. When the larger set is not found in the discourse, the relative clause becomes incoherent given the context, causing reading difficulty. Modeling this coherence constraint is essentially a pragmatics problem, and is under the purview of the pragmatics processor in our system. The pragmatics processor is quite specialised and, although the information it encapsulates is quite intuitive, it nonetheless relies on hand-coded expert knowledge.

The pragmatics processor takes as input an incremental pragmatics configuration p and computes the probability $P_{prag}(p|w,t,c)$. The pragmatics configuration we consider is quite simple. It is a 3-tuple where one element is true if the current noun phrase being processed is a discourse new definite noun phrase, the second

element is true if the current NP is a discourse new indefinite noun phrase, and the final element is true if we encounter an unsupported restrictive relative clause. We simply conjecture that there is little processing cost (and hence a high probability) if the entire vector is false; there is a small processing cost for discourse new indefinites, a slightly larger processing cost for discourse new definites and a large processing cost for an incoherent reduced relative clause.

The first two elements of the 3-tuple depend on the identity of the determiner as recovered by the parser, and on whether the coreference system adduces the predicate $First$ for the current NP. As the coreference system wasn't designed to find anaphoric contrast sets, these sets were found using a simple post-processing check. This post-processing approach worked well for our experimental items, but finding such sets is, in general, quite a difficult problem (Modjeska et al., 2003).

The distribution $P_{prag}(p|w,t,c)$ applies a processing penalty for an unsupported restrictive relative clause whenever a restrictive relative clause is in the n best list. Because Surprisal computes a ratio of probabilities, this in effect means we only pay this penality when an unsupported restrictive relative clause first appears in the n best list (otherwise the effect is cancelled out). The penalty for discourse new entities is applied on the first word (ignoring punctuation) following the end of the NP. This spillover processing effect is simply a matter of modeling convenience: without it, we would have to compute Surprisal probabilities over regions rather than individual words. Thus, the overall prefix probability can be computed as: $P(w) = \sum_{p,c,t} P_{prag}(p|w,t,c)P_{coref}(c|w,t)P_{parser}(w,t)$, which is then substituted in Equation (1) to get a Surprisal prediction for the current word.

4 Evaluation

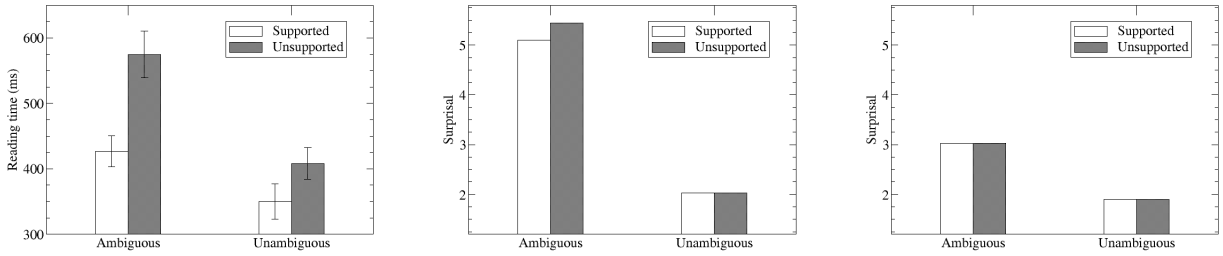
4.1 Method

When modeling the garden path experiment we presented in Section 2.1, we compute Surprisal values on the word 'by', which is the earliest point at which there is evidence for a relative clause interpretation. For the Grodner et al. experiment, we compute Surprisal values on the relativiser 'who' or 'that'. Again, this is the earliest point at which there is evidence for a relative clause, and depending upon the presence or absence of a preceding comma, it will be known to be restrictive or nonrestrictive clause. In addition to the overall Surprisal values, we also compute syntactic Surprisal scores, to test if there is any benefit from the discourse and pragmatics subsystems. As we are outputting n best lists for each parse, it is also straightforward to compute other measures which predict reading difficulty, including pruning (Jurafsky, 1996), whereby processing difficulty is predicted when a parse is removed from the n best list, and attention shift (Crocker and Brants, 2000), which predicts parsing difficulty at words where the most highly ranked parse flips from one interpretation to another.

For the garden path experiment, the simulation was run on each of the 28 experimental items in each of the 4 conditions, resulting in a total of 112 runs. For the Grodner et al. experiment, the simulation was run on each of the 20 items in each of the 4 conditions, resulting in a total of 80 runs. For each run, the model was reset, purging all discourse information gained while reading earlier items. As the system is not stochastic, two runs using the exact same items in the same condition will produce the same result. Therefore, we made no attempt to model by-subject variability, but we did perform by-item ANOVAs on the system output.

4.2 Results

Garden Path Experiment The simulated results of our experiment are shown in Figure 2. Comparing the full simulated results in Figure 2b to the experimental results in Figure 2a, we find that the simulation, like the actual experiment, finds both main effects and an interaction: there is a main effect of context whereby a supportive context facilitates reading, a main effect of syntax whereby the garden path slows down reading, and an interaction in that the effect of context is strongest in the garden path condition. All these effects were highly significant at $p < 0.01$. The pattern of results between the full simulation and the experiment differed in two ways. First, the simulated results suggested a much larger reading difficulty due to ambiguity than the experimental results. Also, in the unambiguous case, the model predicted a null cost of an unsupportive context on the word 'by', because the model bears the cost of an unsupportive context earlier in the sentence, and assumes no spillover to the word 'by'. Finally, we note that the syntax-only simulation, shown in Figure 2c, only produced a main effect of ambigu-

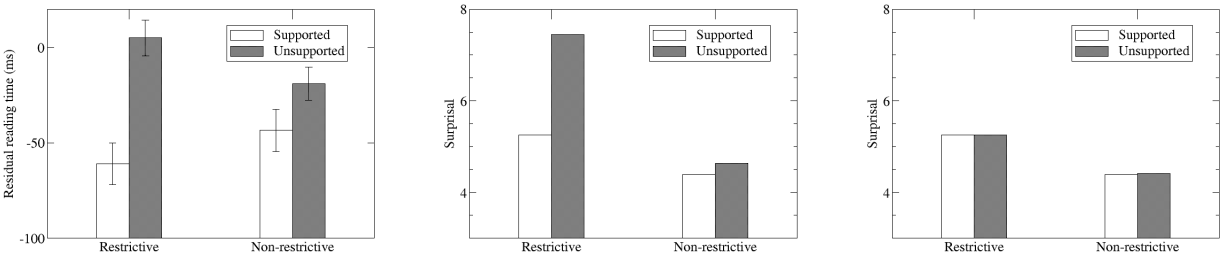


(a) Results from our garden path experiment

(b) Simulation of our garden path experiment

(c) Syntax-only simulation

Figure 2: The simulated results predict the same interaction as the garden path experiment, but show a stronger main effect of ambiguity, and no influence of discourse in the unambiguous condition on the word 'by'.



(a) Results from the Grodner et al. experiment

(b) Simulation of the Grodner et al. experiment

(c) Syntax-only simulation

Figure 3: The simulated results predict the outcome of the Grodner et al. experiment.

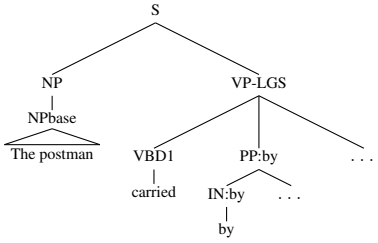
ity, and was not able to model the effect of context.

Grodner et al. Experiment The simulated results of the Grodner et al. experiment are shown in Figure 3. In this experiment, the pattern of simulated results in Figure 3b showed a much closer resemblance to the experimental results in Figure 3a than the garden path experiment. There is a main effect of context, which is much stronger in the restrictive relative case compared to non-restrictive relatives. As with the garden path experiment, the ANOVA reported that all effects were significant at the $p < 0.01$ level. Again, as we can see from Figure 3c, there was no effect of context in the syntax-only simulation. The numerical trend did show a slight facilitation in the unrestricted supported condition, with a Surprisal of 4.39 compared to 4.41 in the supported case, but this difference was not significant.

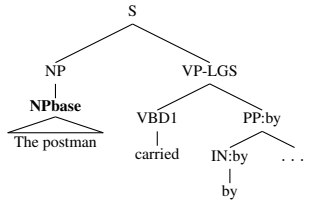
4.3 Discussion

We have shown that our incremental sentence processor augmented with discourse processing can successfully simulate syntax-discourse interaction effects which have been shown in the literature. The difference between a Weakly Interactive and Strongly Interactive model can be thought of computationally in terms of a pipeline architecture versus joint inference. In a weaker sense, even a pipeline architecture where the discourse can influence syntactic probabilities could be claimed to be a Strongly Interactive model. However, as our model uses a pipeline where syntactic probabilities are independent of the discourse, we claim that our model is Weakly Interactive. Unlike Altmann and Steedman, who posited that the discourse processor actually removes parsing hypotheses, we were able to simulate this pruning behaviour by simply re-weighting parses in our coreference and pragmatics modules.

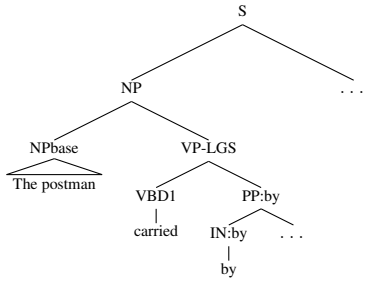
The fact that a Weakly Interactive system can simulate the result of an experiment proposed in support of the Strongly Interactive hypothesis is initially counter-intuitive. However, this naturally falls out from our decision to use a probabilistic



(a) Best parse: $p = 9.99 \times 10^{-10}$ main clause, expecting more dependents



(b) 2^{nd} parse: $p = 9.93 \times 10^{-10}$ main clause, no more dependents



(c) 3^{rd} parse: $p = 7.69\times^{-10}$ relative clause

Figure 4: The top three parses on the word 'by' in the our first experimental item.

model: a lower probability, even in an unambiguous structure, is associated with increased reading difficulty. As an aside, we note that when using realistic computational grammars, even the structures used in the Grodner et al. experiment are not unambiguous. In the restrictive relative clause condition, even though there was not any competition between a relative and main clause reading, our n best list was at all times filled with analyses. For example, on the word 'who' in the restricted relative clause condition, the parser is already predicting both the subject-relative ('the postman who was bit by the dog') and object-relative ('the postman who the dog bit') readings.

Overall, these results are supportive of the growing importance of probabilistic reasoning as a model of human cognitive behaviour. Therefore, especially with respect to sentence processing, it is necessary to have a proper understanding of how probabilities are linked to real-world behaviours. We note that Surprisal does indeed show processing difficulty on the word 'by' in the garden path experiment. However, Figure 4 (which shows the top three parses on the word 'by') indicates that not only are there still main clause interpretations present, but in fact, the top two parses are main clause interpretations. This is also true if we limit ourselves to syntactic probabilities (which are the probabilities listed in Figure 4). This suggests that neither Jurafsky (1996)'s notion of pruning as processing difficulty nor Crocker and Brants (2000) notion of attention shifts would correctly predict higher reading times on a region containing the word 'by'. In fact, the main clause interpretation remains the highest-ranked interpretation until it is finally pruned at an auxiliary of the main verb of the sentence ('The postman carried by the paramedics was *having*').

This result is curious as our experimental items closely match some of those simulated by Crocker and Brants (2000). We conjecture that the difference between our attention shift prediction and theirs is due to differences in the grammar. It is possible that using a more highly tuned grammar would result in attention shift making the correct prediction, but this possibly shows one benefit of using Surprisal as a linking hypothesis. Because Surprisal sums over several derivations, it is not as reliant upon the grammar as the attention shift or pruning linking hypotheses.

5 Conclusions

The main result of this paper is that it is possible to produce a Surprisal-based sentence processing model which can simulate the influence of discourse on syntax in both garden path and unambiguous sentences. Computationally, the inclusion of Markov Logic allowed the discourse module to compute well-formed coreference chains, and opens two avenues of future research. First, it ought to be possible to make the probabilistic logic more naturally incremental, rather than re-running from scratch at each word. Second, we would like to make greater use of the logical elements by applying it to problems where inference is necessary, such as resolving bridging anaphora (Haviland and Clark, 1974).

Our primary cognitive finding that our model, which assumes the Weakly Interactive hypothesis (whereby discourse is influenced by syntax in a reactive manner), is nonetheless able to simulate the experimental results of Grodner et al. (2005), which were claimed by the authors to be in

support of the Strongly Interactive hypothesis. This suggests that the evidence is in favour of the Strongly Interactive hypothesis may be weaker than thought.

Finally, we found that the attention shift (Crocker and Brants, 2000) and pruning (Jurafsky, 1996) linking theories are unable to correctly simulate the results of the garden path experiment. Although our main results above underscore the usefulness of probabilistic modeling, this observation emphasizes the importance of finding a tenable link between probabilities and behaviours.

Acknowledgements

We would like to thank Frank Keller, Patrick Sturt, Alex Lascarides, Mark Steedman, Mirella Lapata and the anonymous reviewers for their insightful comments. We would also like to thank ESRC for their financial supporting on grant RES-062-23-1450.

References

Gerry Altmann and Mark Steedman. Interaction with context during human sentence processing. *Cognition*, 30:191–238, 1988.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference (LREC 98)*, 1998.

Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl, and Shravan Vasisht. Surprising parser actions and reading difficulty. In *Proceedings of ACL-08:HLT, Short Papers*, pages 5–8, 2008.

Thorsten Brants and Matthew Crocker. Probabilistic parsing and psychological plausibility. In *Proceedings of 18th International Conference on Computational Linguistics (COLING-2000)*, pages 111–117, 2000.

Stephen Crain and Mark Steedman. On not being led down the garden path: the use of context by the psychological syntax processor. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural language parsing: Psychological, computational, and theoretical perspectives*. Cambridge University Press, 1985.

Matthew Crocker and Thorsten Brants. Wide coverage probabilistic sentence processing. *Journal of Psycholinguistic Research*, 29(6): 647–669, 2000.

Vera Demberg and Frank Keller. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of the 29th meeting of the Cognitive Science Society (CogSci-09)*, 2009.

Amit Dubey, Frank Keller, and Patrick Sturt. A probabilistic corpus-based model of parallelism. *Cognition*, 109(2):193–210, 2009.

Amit Dubey, Patrick Sturt, and Frank Keller. The effect of discourse inferences on syntactic ambiguity resolution. In *Proceedings of the 23rd Annual CUNY Conference on Human Sentence Processing (CUNY 2010)*, page 151, 2010.

Ted Gibson. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76, 1998.

Daniel J. Grodner, Edward A. F. Gibson, and Duane Watson. The influence of contextual constrast on syntactic processing: Evidence for strong-interaction in sentence comprehension. *Cognition*, 95(3):275–296, 2005.

John T. Hale. A probabilistic earley parser as a psycholinguistic model. In *In Proceedings of the Second Meeting of the North American Chapter of the Asssociation for Computational Linguistics*, 2001.

John T. Hale. The information conveyed by words in sentences. *Journal of Psycholinguistic Research*, 32(2):101–123, 2003.

Susan E. Haviland and Herbert H. Clark. What's new? acquiring new information as a process in comprehension. *Journal of Verbal Learning and Verbal Behavior*, 13:512–521, 1974.

Shujian Huang, Yabing Zhang, Junsheng Zhou, and Jiajun Chen. Coreference resolution using markov logic. In *Proceedings of the 2009 Conference on Intelligent Text Processing and Computational Linguistics (CICLing 09)*, 2009.

D. Jurafsky. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194, 1996.

Roger Levy. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177, March 2008.

Roger Levy and T. Florian Jaeger. Speakers optimize information density through syntactic reduction. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, 2007.

Ken McRae, Michael J. Spivey-Knowlton, and Michael K. Tanenhaus. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312, 1998.

Don C. Mitchell, Martin M. B. Corley, and Alan Garnham. Effects of context in human sentence parsing: Evidence against a discourse-baed proposal mechanism. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 18(1):69–88, 1992.

Natalia N. Modjeska, Katja Markert, and Malvina Nissim. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pages 176–183, Sapporo, Japan, 2003.

Shrini Narayanan and Daniel Jurafsky. Bayesian models of human sentence processing. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society (CogSci 98)*, 1998.

Ulrike Padó, Matthew Crocker, and Frank Keller. Modelling semantic role plausability in human sentence processing. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci 2006)*, pages 657–662, 2006.

Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, 2008.

Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422, 1998.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62 (1-2):107–136, 2006.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

M. J. Spivey and M. K. Tanenhaus. Syntactic ambiguity resolution in discourse: Modeling the effects of referential context and lexical frequency. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 24(6): 1521–1543, 1998.

Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.

Complexity Metrics in an Incremental Right-corner Parser

Stephen Wu **Asaf Bachrach**[†] **Carlos Cardenas**[*] **William Schuler**[°]
Department of Computer Science, University of Minnesota
[†] Unit de Neuroimagerie Cognitive INSERM-CEA
[*] Department of Brain & Cognitive Sciences, Massachussetts Institute of Technology
[°] University of Minnesota and The Ohio State University
swu@cs.umn.edu [†]asaf@mit.edu [*]cardenas@mit.edu [°]schuler@ling.ohio-state.edu

Abstract

Hierarchical HMM (HHMM) parsers make promising cognitive models: while they use a bounded model of working memory and pursue incremental hypotheses in parallel, they still achieve parsing accuracies competitive with chart-based techniques. This paper aims to validate that a right-corner HHMM parser is also able to produce *complexity metrics*, which quantify a reader's incremental difficulty in understanding a sentence. Besides defining standard metrics in the HHMM framework, a new metric, *embedding difference*, is also proposed, which tests the hypothesis that HHMM store elements represents syntactic working memory. Results show that HHMM surprisal outperforms all other evaluated metrics in predicting reading times, and that embedding difference makes a significant, independent contribution.

1 Introduction

Since the introduction of a parser-based calculation for surprisal by Hale (2001), statistical techniques have been become common as models of reading difficulty and linguistic complexity. Surprisal has received a lot of attention in recent literature due to nice mathematical properties (Levy, 2008) and predictive ability on eye-tracking movements (Demberg and Keller, 2008; Boston et al., 2008a). Many other complexity metrics have been suggested as mutually contributing to reading difficulty; for example, entropy reduction (Hale, 2006), bigram probabilities (McDonald and Shillcock, 2003), and split-syntactic/lexical versions of other metrics (Roark et al., 2009).

A parser-derived complexity metric such as surprisal can only be as good (empirically) as the model of language from which it derives (Frank, 2009). Ideally, a psychologically-plausible language model would produce a surprisal that would correlate better with linguistic complexity. Therefore, the specification of how to encode a syntactic language model is of utmost importance to the quality of the metric.

However, it is difficult to quantify linguistic complexity and reading difficulty. The two commonly-used empirical quantifications of reading difficulty are eye-tracking measurements and word-by-word reading times; this paper uses reading times to find the predictiveness of several parser-derived complexity metrics. Various factors (i.e., from syntax, semantics, discourse) are likely necessary for a full accounting of linguistic complexity, so current computational models (with some exceptions) narrow the scope to syntactic or lexical complexity.

Three complexity metrics will be calculated in a Hierarchical Hidden Markov Model (HHMM) parser that recognizes trees in right-corner form (the left-right dual of left-corner form). This type of parser performs competitively on standard parsing tasks (Schuler et al., 2010); also, it reflects plausible accounts of human language processing as incremental (Tanenhaus et al., 1995; Brants and Crocker, 2000), as considering hypotheses probabilistically in parallel (Dahan and Gaskell, 2007), as bounding memory usage to short-term memory limits (Cowan, 2001), and as requiring more memory storage for center-embedding structures than for right- or left-branching ones (Chomsky and Miller, 1963; Gibson, 1998). Also, unlike most other parsers, this parser preserves the *arc-eager/arc-standard* ambiguity of Abney and John-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1189–1198,
Uppsala, Sweden, 11-16 July 2010.

son (1991). Typical parsing strategies are arc-standard, keeping all right-descendants open for subsequent attachment; but since there can be an unbounded number of such open constituents, this assumption is not compatible with simple models of bounded memory. A consistently arc-eager strategy acknowledges memory bounds, but yields dead-end parses. Both analyses are considered in right-corner HHMM parsing.

The purpose of this paper is to determine whether the language model defined by the HHMM parser can also predict reading times — it would be strange if a psychologically plausible model did not also produce viable complexity metrics. In the course of showing that the HHMM parser does, in fact, predict reading times, we will define surprisal and entropy reduction in the HHMM parser, and introduce a third metric called *embedding difference*.

Gibson (1998; 2000) hypothesized two types of syntactic processing costs: *integration cost*, in which incremental input is combined with existing structures; and *memory cost*, where unfinished syntactic constructions may incur some short-term memory usage. HHMM surprisal and entropy reduction may be considered forms of integration cost. Though typical PCFG surprisal has been considered a forward-looking metric (Demberg and Keller, 2008), the incremental nature of the right-corner transform causes surprisal and entropy reduction in the HHMM parser to measure the likelihood of grammatical structures that were hypothesized before evidence was observed for them. Therefore, these HHMM metrics resemble an integration cost encompassing both backward-looking and forward-looking information.

On the other hand, embedding difference is designed to model the cost of storing center-embedded structures in working memory. Chen, Gibson, and Wolf (2005) showed that sentences requiring more syntactic memory during sentence processing increased reading times, and it is widely understood that center-embedding incurs significant syntactic processing costs (Miller and Chomsky, 1963; Gibson, 1998). Thus, we would expect for the usage of the center-embedding memory store in an HHMM parser to correlate with reading times (and therefore linguistic complexity).

The HHMM parser processes syntactic constructs using a bounded number of store states, defined to represent short-term memory elements; additional states are utilized whenever center-embedded syntactic structures are present. Similar models such as Crocker and Brants (2000) implicitly allow an infinite memory size, but Schuler et al. (2008; 2010) showed that a right-corner HHMM parser can parse most sentences in English with 4 or fewer center-embedded-depth levels. This behavior is similar to the hypothesized size of a human short-term memory store (Cowan, 2001). A positive result in predicting reading times will lend additional validity to the claim that the HHMM parser's bounded memory corresponds to bounded memory in human sentence processing.

The rest of this paper is organized as follows: Section 2 defines the language model of the HHMM parser, including definitions of the three complexity metrics. The methodology for evaluating the complexity metrics is described in Section 3, with actual results in Section 4. Further discussion on results, and comparisons to other work, are in Section 5.

2 Parsing Model

This section describes an incremental parser in which surprisal and entropy reduction are simple calculations (Section 2.1). The parser uses a Hierarchical Hidden Markov Model (Section 2.2) and recognizes trees in a right-corner form (Section 2.3 and 2.4). The new complexity metric, embedding difference (Section 2.5), is a natural consequence of this HHMM definition. The model is equivalent to previous HHMM parsers (Schuler, 2009), but reorganized into 5 cases to clarify the right-corner structure of the parsed sentences.

2.1 Surprisal and Entropy in HMMs

Hidden Markov Models (HMMs) probabilistically connect sequences of observed states o_t and hidden states q_t at corresponding time steps t. In parsing, observed states are words; hidden states can be a conglomerate state of linguistic information, here taken to be syntactic.

The HMM is an incremental, time-series structure, so one of its by-products is the *prefix probability*, which will be used to calculate surprisal. This is the probability that that words $o_{1..t}$ have been observed at time t, regardless of which syntactic states $q_{1..t}$ produced them. Bayes' Law and Markov independence assumptions allow this to

be calculated from two generative probability distributions.[1]

$$\mathsf{Pre}(o_{1..t}) = \sum_{q_{1..t}} \mathsf{P}(o_{1..t}\, q_{1..t}) \quad (1)$$

$$\stackrel{\text{def}}{=} \sum_{q_{1..t}} \prod_{\tau=1}^{t} \mathsf{P}_{\Theta_\mathrm{A}}(q_\tau \mid q_{\tau-1}) \cdot \mathsf{P}_{\Theta_\mathrm{B}}(o_\tau \mid q_\tau) \quad (2)$$

Here, probabilities arise from a *Transition Model* (Θ_A) between hidden states and an *Observation Model* (Θ_B) that generates an observed state from a hidden state. These models are so termed for historical reasons (Rabiner, 1990).

Surprisal (Hale, 2001) is then a straightforward calculation from the prefix probability.

$$\text{Surprisal}(t) = \log_2 \frac{\mathsf{Pre}(o_{1..t-1})}{\mathsf{Pre}(o_{1..t})} \quad (3)$$

This framing of prefix probability and surprisal in a time-series model is equivalent to Hale's (2001; 2006), assuming that $q_{1..t} \in \mathcal{D}_t$, i.e., that the syntactic states we are considering form derivations $\mathcal{D}_t$, or partial trees, consistent with the observed words. We will see that this is the case for our parser in Sections 2.2–2.4.

Entropy is a measure of uncertainty, defined as $H(x) = -\mathsf{P}(x) \log_2 \mathsf{P}(x)$. Now, the entropy H_t of a t-word string $o_{1..t}$ in an HMM can be written:

$$H_t = \sum_{q_{1..t}} \mathsf{P}(q_{1..t}\, o_{1..t}) \log_2 \mathsf{P}(q_{1..t}\, o_{1..t}) \quad (4)$$

and *entropy reduction* (Hale, 2003; Hale, 2006) at the t^{th} word is then

$$\text{ER}(o_t) = \max(0, H_{t-1} - H_t) \quad (5)$$

Both of these metrics fall out naturally from the time-series representation of the language model. The third complexity metric, embedding difference, will be discussed after additional background in Section 2.5.

In the implementation of an HMM, candidate states at a given time q_t are kept in a trellis, with step-by-step backpointers to the highest-probability $q_{1..t-1}$.[2] Also, the best q_t are often kept in a beam $\mathcal{B}_t$, discarding low-probability states. This mitigates the problems of large state spaces (e.g., that of all possible grammatical derivations). Since beams have been shown to perform well (Brants and Crocker, 2000; Roark, 2001; Boston et al., 2008b), complexity metrics in this paper are calculated on a beam rather than over all (unbounded) possible derivations $\mathcal{D}_t$. The equations above, then, will replace the assumption $q_{1..t} \in \mathcal{D}_t$ with $q_t \in \mathcal{B}_t$.

[1] Technically, a prior distribution over hidden states, $\mathsf{P}(q_0)$, is necessary. This q_0 is factored and taken to be a deterministic constant, and is therefore unimportant as a probability model.

[2] Typical tasks in an HMM include finding the most likely sequence via the Viterbi algorithm, which stores these backpointers to maximum-probability previous states and can uniquely find the most likely sequence.

2.2 Hierarchical Hidden Markov Models

Hidden states q can have internal structure; in Hierarchical HMMs (Fine et al., 1998; Murphy and Paskin, 2001), this internal structure will be used to represent syntax trees and looks like several HMMs stacked on top of each other. As such, q_t is factored into sequences of depth-specific variables — one for each of D levels in the HMM hierarchy. In addition, an intermediate variable f_t is introduced to interface between the levels.

$$q_t \stackrel{\text{def}}{=} \langle q_t^1 \ldots q_t^D \rangle \quad (6)$$

$$f_t \stackrel{\text{def}}{=} \langle f_t^1 \ldots f_t^D \rangle \quad (7)$$

Transition probabilities $\mathsf{P}_{\Theta_\mathrm{A}}(q_t \mid q_{t-1})$ over complex hidden states q_t are calculated in two phases:

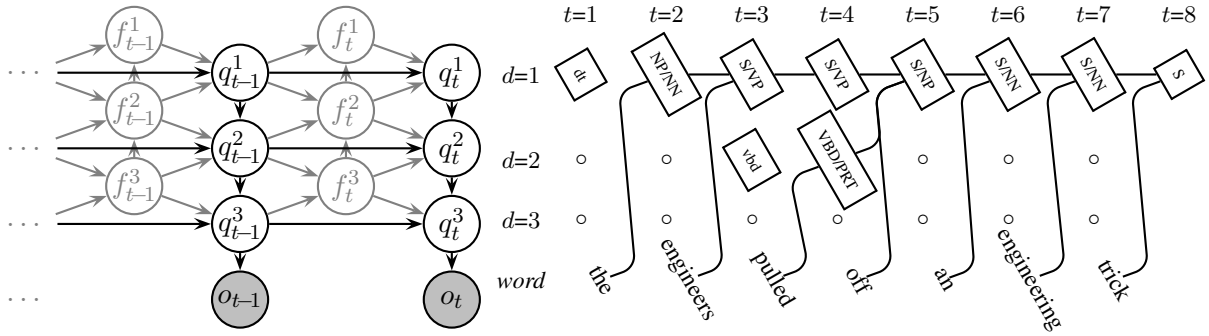
- *Reduce phase.* Yields an intermediate state f_t, in which component HMMs may terminate. This f_t tells "higher" HMMs to hold over their information if "lower" levels are in operation at any time step t, and tells lower HMMs to signal when they're done.
- *Shift phase.* Yields a modeled hidden state q_t, in which unterminated HMMs transition, and terminated HMMs are re-initialized from their parent HMMs.

Each phase is factored according to *level-specific* reduce and shift models, Θ_F and Θ_Q:

$$\mathsf{P}_{\Theta_\mathrm{A}}(q_t|q_{t-1}) = \sum_{f_t} \mathsf{P}(f_t|q_{t-1}) \cdot \mathsf{P}(q_t|f_t\, q_{t-1}) \quad (8)$$

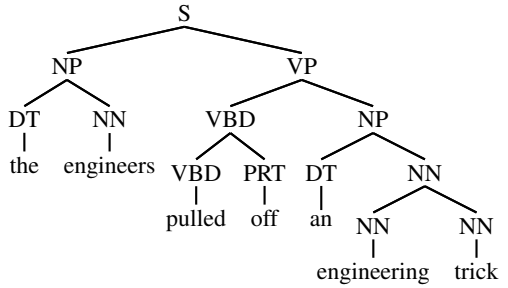
$$\stackrel{\text{def}}{=} \sum_{f_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_\mathrm{F}}(f_t^d|f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \cdot \mathsf{P}_{\Theta_\mathrm{Q}}(q_t^d|f_t^{d+1} f_t^d\, q_{t-1}^d q_t^{d-1}) \quad (9)$$

with f_t^{D+1} and q_t^0 defined as constants. Note that only q_t is present at the end of the probability calculation. In step t, f_{t-1} will be unused, so the marginalization of Equation 9 does not lose any information.

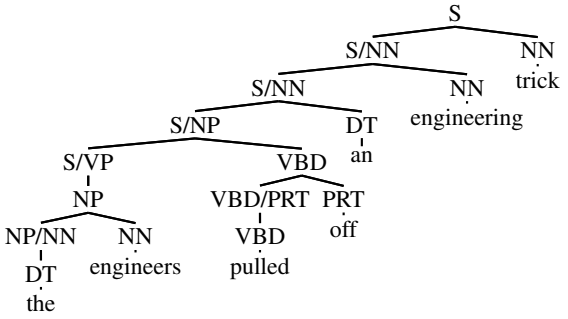


(a) Dependency structure in the HHMM parser. Conditional probabilities at a node are dependent on incoming arcs.

(b) HHMM parser as a store whose elements at each time step are listed vertically, showing a good hypothesis on a sample sentence out of many kept in parallel. Variables corresponding to q_t^d are shown.



(c) A sample sentence in CNF.



(d) The right-corner transformed version of (c).

Figure 1: Various graphical representations of HHMM parser operation. (a) shows probabilistic dependencies. (b) considers the q_t^d store to be incremental syntactic information. (c)–(d) demonstrate the right-corner transform, similar to a left-to-right traversal of (c). In 'NP/NN' we say that NP is the *active* constituent and NN is the *awaited*.

The Observation Model Θ_{B} is comparatively much simpler. It is only dependent on the syntactic state at D (or the deepest active HHMM level).

$$\mathsf{P}_{\Theta_{\mathrm{B}}}(o_t \mid q_t) \stackrel{\text{def}}{=} \mathsf{P}(o_t \mid q_t^D) \tag{10}$$

Figure 1(a) gives a schematic of the dependency structure of Equations 8–10 for $D = 3$. Evaluations in this paper are done with $D = 4$, following the results of Schuler, et al. (2008).

2.3 Parsing right-corner trees

In this HHMM formulation, states and dependencies are optimized for parsing right-corner trees (Schuler et al., 2008; Schuler et al., 2010). A sample transformation between CNF and right-corner trees is in Figures 1(c)–1(d).

Figure 1(b) shows the corresponding store-element interpretation[3] of the right corner tree in 1(d). These can be used as a case study to see what kind of operations need to occur in an HHMM when parsing right-corner trees. There is one unique set of HHMM state values for each tree, so the operations can be seen on either the tree or the store elements.

At each time step t, a certain number of elements (maximum D) are kept in memory, i.e., in the store. New words are observed input, and the bottom occupied element (the "frontier" of the store) is the context; together, they determine what the store will look like at $t{+}1$. We can characterize the types of store-element changes by when they happen in Figures 1(b) and 1(d):

Cross-level Expansion (CLE). Occupies a new store element at a given time step. For example, at $t = 1$, a new store element is occupied which can interact with the observed word, "the." At $t = 3$, an expansion occupies the second store element.

In-level Reduction (ILR). Completes an active constituent that is a unary child in the right-corner tree; always accompanied by an in-level expansion. At $t = 2$, "engineers" completes the active NP constituent; however, the

[3] This is technically a pushdown automoton (PDA), where the store is limited to D elements. When referring to directions (e.g., up, down), PDAs are typically described opposite of the one in Figure 1(b); here, we push "up" instead of down.

level is not yet complete since the NP is along the left-branching trunk of the tree.

In-level Expansion (ILE). Starts a new active constituent at an already-occupied store element; always follows an in-level reduction. With the NP complete in $t=2$, a new active constituent S is produced at $t=3$.

In-level Transition (ILT). Transitions the store to a new state in the next time step at the same level, where the awaited constituent changes and the active constituent remains the same. This describes each of the steps from $t=4$ to $t=8$ at $d=1$.

Cross-level Reduction (CLR). Vacates a store element on seeing a complete active constituent. This occurs after $t = 4$; "off" completes the active (at depth 2) VBD constituent, and vacates store element 2. This is accompanied with an in-level transition at depth 1, producing the store at $t=5$. It should be noted that with some probability, completing the active constituent does not vacate the store element, and the in-level reduction case would have to be invoked.

The in-level/cross-level ambiguity occurs in the expansion as well as the reduction, similar to Abney and Johnson's arc-eager/arc-standard composition strategies (1991). At $t=3$, another possible hypothesis would be to remain on store element 1 using an ILE instead of a CLE. The HHMM parser, unlike most other parsers, will preserve this in-level/cross-level ambiguity by considering both hypotheses in parallel.

2.4 Reduce and Shift Models

With the understanding of what operations need to occur, a formal definition of the language model is in order. Let us begin with the relevant variables.

A shift variable q_t^d at depth d and time step t is a syntactic state that must represent the active and awaited constituents of right-corner form:

$$q_t^d \stackrel{\text{def}}{=} \langle g^{\text{A}}_{q_t^d}, g^{\text{W}}_{q_t^d} \rangle \tag{11}$$

e.g., in Figure 1(b), q_2^1=⟨NP,NN⟩=NP/NN. Each g is a constituent from the pre-right-corner grammar, G.

Reduce variables f are then enlisted to ensure that in-level and cross-level operations are correct.

$$f_t^d \stackrel{\text{def}}{=} \langle k_{f_t^d}, g_{f_t^d} \rangle \tag{12}$$

First, $k_{f_t^d}$ is a switching variable that differentiates between ILT, CLE/CLR, and ILE/ILR. This switching is the most important aspect of f_t^d, so regardless of what $g_{f_t^d}$ is, we will use:

- $f_t^d \in F_0$ when $k_{f_t^d} = \mathbf{0}$, (ILT/no-op)
- $f_t^d \in F_1$ when $k_{f_t^d} = \mathbf{1}$, (CLE/CLR)
- $f_t^d \in F_G$ when $k_{f_t^d} \in G$. (ILE/ILR)

Then, $g_{f_t^d}$ is used to keep track of a completely-recognized constituent whenever a reduction occurs (ILR or CLR). For example, in Figure 1(b), after time step 2, an NP has been completely recognized and precipitates an ILR. The NP gets stored in $g_{f_3^1}$ for use in the ensuing ILE instead of appearing in the store-elements.

This leads us to a specification of the reduce and shift probability models. The reduce step happens first at each time step. True to its name, the reduce step handles in-level and cross-level reductions (the second and third case below):

$$\mathsf{P}_{\Theta_{\text{F}}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \stackrel{\text{def}}{=}$$
$$\begin{cases} \text{if } f_t^{d+1} \notin F_G & : [\![f_t^d = \mathbf{0}]\!] \\ \text{if } f_t^{d+1} \in F_G, f_t^d \in F_1 & : \tilde{\mathsf{P}}_{\Theta_{\text{F-ILR},d}}(f_t^d \mid q_{t-1}^d \, q_{t-1}^{d-1}) \\ \text{if } f_t^{d+1} \in F_G, f_t^d \in F_G & : \tilde{\mathsf{P}}_{\Theta_{\text{F-CLR},d}}(f_t^d \mid q_{t-1}^d \, q_{t-1}^{d-1}) \end{cases} \tag{13}$$

with edge cases q_t^0 and f_t^{D+1} defined as appropriate constants. The first case is just store-element maintenance, in which the variable is not on the "frontier" and therefore inactive.

Examining $\Theta_{\text{F-ILR},d}$ and $\Theta_{\text{F-CLR},d}$, we see that the produced f_t^d variables are also used in the "if" statement. These models can be thought of as picking out a f_t^d first, finding the matching case, then applying the probability models that matches. These models are actually two parts of the same model when learned from trees.

Probabilities in the shift step are also split into cases based on the reduce variables. More maintenance operations (first case) accompany transitions producing new awaited constituents (second case below) and expansions producing new active constituents (third and fourth case):

$$\mathsf{P}_{\Theta_{\text{Q}}}(q_t^d \mid f_t^{d+1} f_t^d \, q_{t-1}^d q_t^{d-1}) \stackrel{\text{def}}{=}$$
$$\begin{cases} \text{if } f_t^{d+1} \notin F_G & : [\![q_t^d = q_{t-1}^d]\!] \\ \text{if } f_t^{d+1} \in F_G, \, f_t^d \in F_0 & : \tilde{\mathsf{P}}_{\Theta_{\text{Q-ILT},d}}(q_t^d \mid f_t^{d+1} \, q_{t-1}^d \, q_t^{d-1}) \\ \text{if } f_t^{d+1} \in F_G, \, f_t^d \in F_1 & : \tilde{\mathsf{P}}_{\Theta_{\text{Q-ILE},d}}(q_t^d \mid f_t^d \, q_{t-1}^d \, q_t^{d-1}) \\ \text{if } f_t^{d+1} \in F_G, \, f_t^d \in F_G & : \tilde{\mathsf{P}}_{\Theta_{\text{Q-CLE},d}}(q_t^d \mid q_t^{d-1}) \end{cases} \tag{14}$$

FACTOR	DESCRIPTION	EXPECTED
Word order in narrative	For each story, words were indexed. Subjects would tend to read faster later in a story.	negative slope
Reciprocal length	Log of the reciprocal of the number of letters in each word. A decrease in the reciprocal (increase in length) might mean longer reading times.	positive slope
Unigram frequency	A log-transformed empirical count of word occurrences in the Brown Corpus section of the Penn Treebank. Higher frequency should indicate shorter reading times.	negative slope
Bigram probability	A log-transformed empirical count of two-successive-word occurrences, with Good-Turing smoothing on words occuring less than 10 times.	negative slope
Embedding difference	Amount of change in HHMM weighted-average embedding depth. Hypothesized to increase with larger working memory requirements, which predict longer reading times.	positive slope
Entropy reduction	Amount of decrease in the HHMM's uncertainty about the sentence. Larger reductions in uncertainty are hypothesized to take longer.	positive slope
Surprisal	"Surprise value" of a word in the HHMM parser; models were trained on the Wall Street Journal, sections 02–21. More surprising words may take longer to read.	positive slope

Table 1: A list of factors hypothesized to contribute to reading times. All data was mean-centered.

A final note: the notation $\tilde{\mathsf{P}}_\Theta(\cdot \mid \cdot)$ has been used to indicate probability models that are empirical, trained directly from frequency counts of right-corner transformed trees in a large corpus. Alternatively, a standard PCFG could be trained on a corpus (or hand-specified), and then the grammar itself can be right-corner transformed (Schuler, 2009).

Taken together, Equations 11–14 define the probabilistic structure of the HHMM for parsing right-corner trees.

2.5 Embedding difference in the HHMM

It should be clear from Figure 1 that at any time step while parsing depth-bounded right-corner trees, the candidate hidden state q_t will have a "frontier" depth $d(q_t)$. At time t, the beam of possible hidden states q_t stores the syntactic state (and a backpointer) along with its probability, $\mathsf{P}(o_{1..t}\, q_{1..t})$. The *average embedding depth* at a time step is then

$$\mu_{\text{EMB}}(o_{1..t}) = \sum_{q_t \in \mathcal{B}_t} d(q_t) \cdot \frac{\mathsf{P}(o_{1..t}\, q_{1..t})}{\sum_{q'_t \in \mathcal{B}_t} \mathsf{P}(o_{1..t}\, q'_{1..t})} \tag{15}$$

where we have directly used the beam notation. The *embedding difference* metric is:

$$\text{EmbDiff}(o_{1..t}) = \mu_{\text{EMB}}(o_{1..t}) - \mu_{\text{EMB}}(o_{1..t-1}) \tag{16}$$

There is a strong computational correspondence between this definition of embedding difference and the previous definition of surprisal. To see this, we rewrite Equations 1 and 3:

$$\mathsf{Pre}(o_{1..t}) = \sum_{q_t \in \mathcal{B}_t} \mathsf{P}(o_{1..t}\, q_{1..t}) \tag{1'}$$

$$\text{Surprisal}(t) = \log_2 \mathsf{Pre}(o_{1..t-1}) - \log_2 \mathsf{Pre}(o_{1..t}) \tag{3'}$$

Both surprisal and embedding difference include summations over the elements of the beam, and are calculated as a difference between previous and current beam states.

Most differences between these metrics are relatively inconsequential. For example, the difference in order of subtraction only assures that a positive correlation with reading times is expected. Also, the presence of a logarithm is relatively minor. Embedding difference weighs the probabilities with center-embedding depths and then normalizes the values; since the measure is a weighted average of embedding depths rather than a probability distribution, μ_{EMB} is not always less than 1 and the correspondence with Kullback-Leibler divergence (Levy, 2008) does not hold, so it does not make sense to take the logs.

Therefore, the inclusion of the embedding depth, $d(q_t)$, is the only significant difference between the two metrics. The result is a metric that, despite numerical correspondence to surprisal, models the HHMM's hypotheses about memory cost.

3 Evaluation

Surprisal, entropy reduction, and embedding difference from the HHMM parser were evaluated against a full array of factors (Table 1) on a corpus of word-by-word reading times using a linear mixed-effects model.

The corpus of reading times for 23 native English speakers was collected on a set of four narratives (Bachrach et al., 2009), each composed of sentences that were syntactically complex but constructed to appear relatively natural. Using Linger 2.88, words appeared one-by-one on the screen, and required a button-press in order to advance; they were displayed in lines with 11.5 words on average.

Following Roark et al.'s (2009) work on the same corpus, reading times above 1500 ms (for diverted attention) or below 150 ms (for button presses planned before the word appeared) were discarded. In addition, the first and last word of each line on the screen were removed; this left 2926 words out of 3540 words in the corpus.

For some tests, a division between open- and closed-class words was made, with 1450 and 1476 words, respectively. Closed-class words (e.g., determiners or auxiliary verbs) usually play some kind of syntactic function in a sentence; our evaluations used Roark et al.'s list of stop words. Open class words (e.g., nouns and other verbs) more commonly include new words. Thus, one may expect reading times to differ for these two types of words.

Linear mixed-effect regression analysis was used on this data; this entails a set of *fixed effects* and another of *random effects*. Reading times y were modeled as a linear combination of factors x, listed in Table 1 (fixed effects); some random variation in the corpus might also be explained by groupings according to subject i, word j, or sentence k (random effects).

$$y_{ijk} = \beta_0 + \sum_{\ell=1}^{m} \beta_\ell x_{ijk\ell} + b_i + b_j + b_k + \varepsilon \qquad (17)$$

This equation is solved for each of m fixed-effect coefficients β with a measure of confidence (t-value $= \hat{\beta}/\mathrm{SE}(\hat{\beta})$, where SE is the standard error). β_0 is the standard intercept to be estimated along with the rest of the coefficients, to adjust for affine relationships between the dependent and independent variables. We report factors as statistically significant contributors to reading time if the absolute value of the t-value is greater than 2.

Two more types of comparisons will be made to see the significance of factors. First, a model of data with the full list of factors can be compared to a model with a subset of those factors. This is done with a likelihood ratio test, producing (for mixed-effects models) a χ^2_1 value and corresponding probability that the smaller model could have produced the same estimates as the larger model. A lower probability indicates that the additional factors in the larger model are significant.

Second, models with different fixed effects can be compared to each other through various information criteria; these trade off between having a more explanatory model vs. a simpler model, and can be calculated on any model. Here, we use Akaike's Information Criterion (AIC), where lower values indicate better models.

All these statistics were calculated in R, using the `lme4` package (Bates et al., 2008).

4 Results

Using the full list of factors in Table 1, fixed-effect coefficients were estimated in Table 2. Fitting the best model by AIC would actually prune away some of the factors as relatively insignificant, but these smaller models largely accord with the significance values in the table and are therefore not presented.

The first data column shows the regression on all data; the second and third columns divide the data into open and closed classes, because an evaluation (not reported in detail here) showed statistically significant interactions between word class and 3 of the predictors. Additionally, this facilitates comparison with Roark et al. (2009), who make the same division.

Out of the non-parser-based metrics, word order and bigram probability are statistically significant regardless of the data subset; though reciprocal length and unigram frequency do not reach significance here, likelihood ratio tests (not shown) confirm that they contribute to the model as a whole. It can be seen that nearly all the slopes have been estimated with signs as expected, with the exception of reciprocal length (which is not statistically significant).

Most notably, HHMM surprisal is seen here to be a standout predictive measure for reading times regardless of word class. If the HHMM parser is a good psycholinguistic model, we would expect it to at least produce a viable surprisal metric, and Table 2 attests that this is indeed the case. Though it seems to be less predictive of open classes, a surprisal-only model has the best AIC (-7804) out of any open-class model. Considering the AIC on the full data, the worst model *with* surprisal

	FULL DATA			OPEN CLASS			CLOSED CLASS		
	Coefficient	Std. Err.	t-value	Coefficient	Std. Err.	t-value	Coefficient	Std. Err.	t-value
(Intcpt)	$-9.340\cdot10^{-3}$	$5.347\cdot10^{-2}$	-0.175	$-1.237\cdot10^{-2}$	$5.217\cdot10^{-2}$	-0.237	$-6.295\cdot10^{-2}$	$7.930\cdot10^{-2}$	-0.794
order	$-3.746\cdot10^{-5}$	$7.808\cdot10^{-6}$	-4.797*	$-3.697\cdot10^{-5}$	$8.002\cdot10^{-6}$	-4.621*	$-3.748\cdot10^{-5}$	$8.854\cdot10^{-6}$	-4.232*
rlength	$-2.002\cdot10^{-2}$	$1.635\cdot10^{-2}$	-1.225	$9.849\cdot10^{-3}$	$1.779\cdot10^{-2}$	0.554	$-2.839\cdot10^{-2}$	$3.283\cdot10^{-2}$	-0.865
unigrm	$-8.090\cdot10^{-2}$	$3.690\cdot10^{-1}$	-0.219	$-1.047\cdot10^{-1}$	$2.681\cdot10^{-1}$	-0.391	$-3.847\cdot10^{+0}$	$5.976\cdot10^{+0}$	-0.644
bigrm	$-2.074\cdot10^{+0}$	$8.132\cdot10^{-1}$	-2.551*	$-2.615\cdot10^{+0}$	$8.050\cdot10^{-1}$	-3.248*	$-5.052\cdot10^{+1}$	$1.910\cdot10^{+1}$	-2.645*
embdiff	$9.390\cdot10^{-3}$	$3.268\cdot10^{-3}$	2.873*	$2.432\cdot10^{-3}$	$4.512\cdot10^{-3}$	0.539	$1.598\cdot10^{-2}$	$5.185\cdot10^{-3}$	3.082*
etrpyrd	$2.753\cdot10^{-2}$	$6.792\cdot10^{-3}$	4.052*	$6.634\cdot10^{-4}$	$1.048\cdot10^{-2}$	0.063	$4.938\cdot10^{-2}$	$1.017\cdot10^{-2}$	4.857*
srprsl	$3.950\cdot10^{-3}$	$3.452\cdot10^{-4}$	11.442*	$2.892\cdot10^{-3}$	$4.601\cdot10^{-4}$	6.285*	$5.201\cdot10^{-3}$	$5.601\cdot10^{-4}$	9.286*

Table 2: Results of linear mixed-effect modeling. Significance (indicated by *) is reported at $p < 0.05$.

	(Intr)	order	rlngth	ungrm	bigrm	emdiff	entrpy
order	.000						
rlength	-.006	-.003					
unigrm	.049	.000	-.479				
bigrm	.001	.005	-.006	-.073			
emdiff	.000	.009	-.049	-.089	.095		
etrpyrd	.000	.003	.016	-.014	.020	-.010	
srprsl	.000	-.008	-.033	-.079	.107	.362	.171

Table 3: Correlations in the full model.

(AIC=-10589) outperformed the best model *without* it (AIC=-10478), indicating that the HHMM surprisal is well worth including in the model regardless of the presence of other significant factors.

HHMM entropy reduction predicts reading times on the full dataset and on closed-class words. However, its effect on open-class words is insignificant; if we compare the model of column 2 against one without entropy reduction, a likelihood ratio test gives $\chi^2_1 = 0.0022, p = 0.9623$ (the smaller model could easily generate the same data).

The HHMM's average embedding difference is also significant except in the case of open-class words — removing embedding difference on open-class data yields $\chi^2_1 = 0.2739, p = 0.6007$. But what is remarkable is that there is any significance for this metric at all. Embedding difference and surprisal were relatively correlated compared to other predictors (see Table 3), which is expected because embedding difference is calculated like a weighted version of surprisal. Despite this, it makes an independent contribution to the full-data and closed-class models. Thus, we can conclude that the average embedding depth component affects reading times — i.e., the HHMM's notion of working memory behaves as we would expect human working memory to behave.

5 Discussion

As with previous work on large-scale parser-derived complexity metrics, the linear mixed-effect models suggest that sentence-level factors are effective predictors for reading difficulty — in these evaluations, better than commonly-used lexical and near-neighbor predictors (Pollatsek et al., 2006; Engbert et al., 2005). The fact that HHMM surprisal outperforms even n-gram metrics points to the importance of including a notion of sentence structure. This is particularly true when the sentence structure is defined in a language model that is psycholinguistically plausible (here, bounded-memory right-corner form).

This accords with an understated result of Boston et al.'s eye-tracking study (2008a): a richer language model predicts eye movements during reading better than an oversimplified one. The comparison there is between phrase structure surprisal (based on Hale's (2001) calculation from an Earley parser), and dependency grammar surprisal (based on Nivre's (2007) dependency parser). Frank (2009) similarly reports improvements in the reading-time predictiveness of unlexicalized surprisal when using a language model that is more plausible than PCFGs.

The difference in predictivity due to word class is difficult to explain. One theory may be that closed-class words are less susceptible to random effects because there is a finite set of them for any language, making them overall easier to predict via parser-derived metrics. Or, we could note that since closed-class words often serve grammatical functions in addition to their lexical content, they contribute more information to parser-derived measures than open-class words. Previous work with complexity metrics on this corpus (Roark et al., 2009) suggests that these explanations only account for part of the word-class variation in the performance of predictors.

Further comparsion to Roark et al. will show other differences, such as the lesser role of word length and unigram frequency, lower overall correlations between factors, and the greater predictivity of their entropy metric. In addition, their metrics are different from ours in that they are designed to tease apart lexical and syntactic contributions to reading difficulty. Their notion of entropy, in particular, estimates Hale's definition of entropy on whole derivations (2006) by isolating the predictive entropy; they then proceed to define separate lexical and syntactic predictive entropies. Drawing more directly from Hale, our definition is a whole-derivation metric based on the conditional entropy of the words, given the root. (The root constituent, though unwritten in our definitions, is always included in the HHMM start state, q_0.)

More generally, the parser used in these evaluations differs from other reported parsers in that it is not lexicalized. One might expect for this to be a weakness, allowing distributions of probabilities at each time step in places not licensed by the observed words, and therefore giving poor probability-based complexity metrics. However, we see that this language model performs well despite its lack of lexicalization. This indicates that lexicalization is not a requisite part of syntactic parser performance with respect to predicting linguistic complexity, corroborating the evidence of Demberg and Keller's (2008) 'unlexicalized' (POS-generating, not word-generating) parser.

Another difference is that previous parsers have produced useful complexity metrics without maintaining arc-eager/arc-standard ambiguity. Results show that including this ambiguity in the HHMM at least does not invalidate (and may in fact improve) surprisal or entropy reduction as reading-time predictors.

6 Conclusion

The task at hand was to determine whether the HHMM could consistently be considered a plausible psycholinguistic model, producing viable complexity metrics while maintaining other characteristics such as bounded memory usage. The linear mixed-effects models on reading times validate this claim. The HHMM can straightforwardly produce highly-predictive, standard complexity metrics (surprisal and entropy reduction). HHMM surprisal performs very well in predicting reading times regardless of word class. Our formulation of entropy reduction is also significant except in open-class words.

The new metric, embedding difference, uses the average center-embedding depth of the HHMM to model syntactic-processing memory cost. This metric can only be calculated on parsers with an explicit representation for short-term memory elements like the right-corner HHMM parser. Results show that embedding difference does predict reading times except in open-class words, yielding a significant contribution independent of surprisal despite the fact that its definition is similar to that of surprisal.

Acknowledgments

Thanks to Brian Roark for help on the reading times corpus, Tim Miller for the formulation of entropy reduction, Mark Holland for statistical insight, and the anonymous reviewers for their input. This research was supported by National Science Foundation CAREER/PECASE award 0447685. The views expressed are not necessarily endorsed by the sponsors.

References

Steven P. Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguistic Research*, 20(3):233–250.

Asaf Bachrach, Brian Roark, Alex Marantz, Susan Whitfield-Gabrieli, Carlos Cardenas, and John D.E. Gabrieli. 2009. Incremental prediction in naturalistic language processing: An fMRI study.

Douglas Bates, Martin Maechler, and Bin Dai. 2008. lme4: Linear mixed-effects models using S4 classes. R package version 0.999375-31.

Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl, U. Patil, and Shravan Vasishth. 2008a. Parsing costs as predictors of reading difficulty: An evaluation using the Potsdam Sentence Corpus. *Journal of Eye Movement Research*, 2(1):1–12.

Marisa Ferrara Boston, John T. Hale, Reinhold Kliegl, and Shravan Vasishth. 2008b. Surprising parser actions and reading difficulty. In *Proceedings of ACL-08: HLT, Short Papers*, pages 5–8, Columbus, Ohio, June. Association for Computational Linguistics.

Thorsten Brants and Matthew Crocker. 2000. Probabilistic parsing and psychological plausibility. In *Proceedings of COLING '00*, pages 111–118.

Evan Chen, Edward Gibson, and Florian Wolf. 2005. Online syntactic storage costs in sentence comprehension. *Journal of Memory and Language*, 52(1):144–169.

Noam Chomsky and George A. Miller. 1963. Introduction to the formal analysis of natural languages. In *Handbook of Mathematical Psychology*, pages 269–321. Wiley.

Nelson Cowan. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.

Matthew Crocker and Thorsten Brants. 2000. Wide-coverage probabilistic sentence processing. *Journal of Psycholinguistic Research*, 29(6):647–669.

Delphine Dahan and M. Gareth Gaskell. 2007. The temporal dynamics of ambiguity resolution: Evidence from spoken-word recognition. *Journal of Memory and Language*, 57(4):483–501.

Vera Demberg and Frank Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

Ralf Engbert, Antje Nuthmann, Eike M. Richter, and Reinhold Kliegl. 2005. SWIFT: A dynamical model of saccade generation during reading. *Psychological Review*, 112:777–813.

Shai Fine, Yoram Singer, and Naftali Tishby. 1998. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32(1):41–62.

Stefan L. Frank. 2009. Surprisal-based comparison between a symbolic and a connectionist model of sentence processing. In *Proc. Annual Meeting of the Cognitive Science Society*, pages 1139–1144.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. In *Image, language, brain: Papers from the first mind articulation project symposium*, pages 95–126.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 159–166, Pittsburgh, PA.

John Hale. 2003. *Grammar, Uncertainty and Sentence Processing*. Ph.D. thesis, Cognitive Science, The Johns Hopkins University.

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30(4):609–642.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Scott A. McDonald and Richard C. Shillcock. 2003. Low-level predictive inference in reading: The influence of transitional probabilities on eye movements. *Vision Research*, 43(16):1735–1751.

George Miller and Noam Chomsky. 1963. Finitary models of language users. In R. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, volume 2, pages 419–491. John Wiley.

Kevin P. Murphy and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proc. NIPS*, pages 833–840, Vancouver, BC, Canada.

Joakim Nivre. 2007. Inductive dependency parsing. *Computational Linguistics*, 33(2).

Alexander Pollatsek, Erik D. Reichle, and Keith Rayner. 2006. Tests of the EZ Reader model: Exploring the interface between cognition and eye-movement control. *Cognitive Psychology*, 52(1):1–56.

Lawrence R. Rabiner. 1990. A tutorial on hidden Markov models and selected applications in speech recognition. *Readings in speech recognition*, 53(3):267–296.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriving lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. *Proceedings of the 2009 Conference on Empirical Methods in Natural Langauge Processing*, pages 324–333.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2008. Toward a psycholinguistically-motivated model of language. In *Proceedings of COLING*, pages 785–792, Manchester, UK, August.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*, 36(1).

William Schuler. 2009. Parsing with a bounded stack using a model-based right-corner transform. In *Proceedings of the North American Association for Computational Linguistics (NAACL '09)*, pages 344–352, Boulder, Colorado.

Michael K. Tanenhaus, Michael J. Spivey-Knowlton, Kathy M. Eberhard, and Julie E. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

"Ask not what Textual Entailment can do for You..."

Mark Sammons V.G.Vinod Vydiswaran Dan Roth
University of Illinois at Urbana-Champaign
{mssammon|vgvinodv|danr}@illinois.edu

Abstract

We challenge the NLP community to participate in a large-scale, distributed effort to design and build resources for developing and evaluating solutions to new and existing NLP tasks in the context of Recognizing Textual Entailment. We argue that the single global label with which RTE examples are annotated is insufficient to effectively evaluate RTE system performance; to promote research on smaller, related NLP tasks, we believe more detailed annotation and evaluation are needed, and that this effort will benefit not just RTE researchers, but the NLP community as a whole. We use insights from successful RTE systems to propose a model for identifying and annotating textual inference phenomena in textual entailment examples, and we present the results of a pilot annotation study that show this model is feasible and the results immediately useful.

1 Introduction

Much of the work in the field of Natural Language Processing is founded on an assumption of semantic compositionality: that there are identifiable, separable components of an unspecified inference process that will develop as research in NLP progresses. Tasks such as Named Entity and coreference resolution, syntactic and shallow semantic parsing, and information and relation extraction have been identified as worthwhile tasks and pursued by numerous researchers. While many have (nearly) immediate application to real world tasks like search, many are also motivated by their potential contribution to more ambitious Natural Language tasks. It is clear that the components/tasks identified so far do not suffice in themselves to solve tasks requiring more complex reasoning and synthesis of information; many other tasks must be solved to achieve human-like performance on tasks such as Question Answering. But there is no clear process for identifying potential tasks (other than consensus by a sufficient number of researchers), nor for quantifying their potential contribution to existing NLP tasks, let alone to Natural Language Understanding.

Recent "grand challenges" such as *Learning by Reading*, *Learning To Read*, and *Machine Reading* are prompting more careful thought about the way these tasks relate, and what tasks must be solved in order to understand text sufficiently well to reliably reason with it. This is an appropriate time to consider a **systematic process for identifying semantic analysis tasks relevant to natural language understanding, and for assessing their potential impact on NLU system performance**.

Research on Recognizing Textual Entailment (RTE), largely motivated by a "grand challenge" now in its sixth year, has already begun to address some of the problems identified above. Techniques developed for RTE have now been successfully applied in the domains of Question Answering (Harabagiu and Hickl, 2006) and Machine Translation (Pado et al., 2009), (Mirkin et al., 2009). The RTE challenge examples are drawn from multiple domains, providing a relatively task-neutral setting in which to evaluate contributions of different component solutions, and RTE researchers have already made incremental progress by identifying sub-problems of entailment, and developing ad-hoc solutions for them.

In this paper we challenge the NLP community to contribute to a joint, long-term effort to identify, formalize, and solve textual inference problems motivated by the Recognizing Textual Entailment setting, in the following ways:

(a) Making the Recognizing Textual Entailment setting a central component of evaluation for

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1199–1208,
Uppsala, Sweden, 11-16 July 2010.

relevant NLP tasks such as NER, Coreference, parsing, data acquisition and application, and others. While many "component" tasks are considered (almost) solved in terms of expected improvements in performance on task-specific corpora, it is not clear that this translates to strong performance in the RTE domain, due either to problems arising from unrelated, unsolved entailment phenomena that co-occur in the same examples, or to domain change effects. The RTE task offers an application-driven setting for evaluating a broad range of NLP solutions, and will reinforce good practices by NLP researchers. The RTE task has been designed specifically to exercise textual inference capabilities, in a format that would make RTE systems potentially useful components in other "deep" NLP tasks such as Question Answering and Machine Translation. [1]

(b) Identifying relevant linguistic phenomena, interactions between phenomena, and their likely impact on RTE/textual inference. Determining the correct label for a single textual entailment example requires human analysts to make many smaller, localized decisions which may depend on each other. A broad, carefully conducted effort to identify and annotate such local phenomena in RTE corpora would allow their distributions in RTE examples to be quantified, and allow evaluation of NLP solutions in the context of RTE. It would also allow assessment of the potential impact of a solution to a specific sub-problem on the RTE task, and of interactions between phenomena. Such phenomena will almost certainly correspond to elements of linguistic theory; but this approach brings a data-driven approach to focus attention on those phenomena that are well-represented in the RTE corpora, and which can be identified with sufficiently close agreement.

(c) Developing resources and approaches that allow more detailed assessment of RTE systems. At present, it is hard to know what specific capabilities different RTE systems have, and hence, which aspects of successful systems are worth emulating or reusing. An evaluation framework that could offer insights into the kinds of sub-problems a given system can reliably solve would make it easier to identify significant advances, and thereby promote more rapid advances through reuse of successful solutions and focus on unresolved problems.

In this paper we demonstrate that Textual Entailment systems are already "interesting", in that they have made significant progress beyond a "smart" lexical baseline that is surprisingly hard to beat (section 2). We argue that Textual Entailment, as an application that clearly requires sophisticated textual inference to perform well, requires the solution of a range of sub-problems, some familiar and some not yet known. We therefore propose RTE as a promising and worthwhile task for large-scale community involvement, as it motivates the study of many other NLP problems in the context of general textual inference.

We outline the limitations of the present model of evaluation of RTE performance, and identify kinds of evaluation that would promote understanding of the way individual components can impact Textual Entailment system performance, and allow better objective evaluation of RTE system behavior without imposing additional burdens on RTE participants. We use this to motivate a large-scale annotation effort to provide data with the mark-up sufficient to support these goals.

To stimulate discussion of suitable annotation and evaluation models, we propose a candidate model, and provide results from a pilot annotation effort (section 3). This pilot study establishes the feasibility of an inference-motivated annotation effort, and its results offer a quantitative insight into the difficulty of the TE task, and the distribution of a number of entailment-relevant linguistic phenomena over a representative sample from the NIST TAC RTE 5 challenge corpus. We argue that such an evaluation and annotation effort can identify relevant subproblems whose solution will benefit not only Textual Entailment but a range of other long-standing NLP tasks, and can stimulate development of new ones. We also show how this data can be used to investigate the behavior of some of the highest-scoring RTE systems from the most recent challenge (section 4).

2 NLP Insights from Textual Entailment

The task of Recognizing Textual Entailment (RTE), as formulated by (Dagan et al., 2006), requires automated systems to identify when a human reader would judge that given one span of text (the Text) and some unspecified (but restricted) world knowledge, a second span of text (the Hy-

[1] The *Parser Training and Evaluation using Textual Entailment* track of SemEval 2 takes this idea one step further, by evaluating performance of an isolated NLP task using the RTE methodology.

Text:	The purchase of LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure.
Hyp 1:	BMI acquired another company.
Hyp 2:	BMI bought LexCorp for $3.4Bn.

Figure 1: **Some representative RTE examples.**

pothesis) is true. The task was extended in (Giampiccolo et al., 2007) to include the additional requirement that systems identify when the Hypothesis contradicts the Text. In the example shown in figure 1, this means recognizing that the Text entails Hypothesis 1, while Hypothesis 2 contradicts the Text. This operational definition of Textual Entailment avoids commitment to any specific knowledge representation, inference method, or learning approach, thus encouraging application of a wide range of techniques to the problem.

2.1 An Illustrative Example

The simple RTE examples in figure 1 (most RTE examples have much longer Texts) illustrate some typical inference capabilities demonstrated by human readers in determining whether one span of text contains the meaning of another.

To recognize that Hypothesis 1 is entailed by the text, a human reader must recognize that "another company" in the Hypothesis can match "LexCorp". She must also identify the nominalized relation "purchase", and determine that "A purchased by B" implies "B acquires A".

To recognize that Hypothesis 2 contradicts the Text, similar steps are required, together with the inference that because the stated purchase price is different in the Text and Hypothesis, but with high probability refers to the same transaction, Hypothesis 2 contradicts the Text.

It could be argued that this particular example might be resolved by simple lexical matching; but it should be evident that the Text can be made lexically very dissimilar to Hypothesis 1 while maintaining the Entailment relation, and that conversely, the lexical overlap between the Text and Hypothesis 2 can be made very high, while maintaining the Contradiction relation. This intuition is borne out by the results of the RTE challenges, which show that lexical similarity-based systems are outperformed by systems that use other, more structured analysis, as shown in the next section.

Rank	System id	Accuracy
1	I	0.735
2	E	0.685
3	H	0.670
4	J	0.667
5	G	0.662
6	B	0.638
7	D	0.633
8	F	0.632
9	A	0.615
9	C	0.615
9	K	0.615
-	Lex	0.612

Table 1: Top performing systems in the RTE 5 2-way task.

	Lex	E	G	H	I	J
Lex	**1.000** (184,183)	**0.667** (157,132)	**0.693** (168,122)	**0.678** (152,136)	**0.660** (165,137)	**0.778** (165,135)
E		**1.000** (224,187)	**0.667** (192,112)	**0.675** (178,131)	**0.673** (201,127)	**0.702** (186,131)
G			**1.000** (247,150)	**0.688** (186,120)	**0.713** (218,115)	**0.745** (198,125)
H				**1.000** (219,183)	**0.705** (194,139)	**0.707** (178,136)
I					**1.000** (260,181)	**0.705** (198,135)
J						**1.000** (224,178)

Table 2: In each cell, top row shows observed agreement and bottom row shows the number of correct (positive, negative) examples on which the pair of systems agree.

2.2 The State of the Art in RTE 5

The outputs for all systems that participated in the RTE 5 challenge were made available to participants. We compared these to each other and to a smart lexical baseline (Do et al., 2010) (lexical match augmented with a WordNet similarity measure, stemming, and a large set of low-semantic-content stopwords) to assess the diversity of the approaches of different research groups. To get the fullest range of participants, we used results from the two-way RTE task. We have anonymized the system names.

Table 1 shows that many participating systems significantly outperform our smart lexical baseline. Table 2 reports the observed agreement between systems and the lexical baseline in terms of the percentage of examples on which a pair of systems gave the same label. The agreement between most systems and the baseline is about 67%, which suggests that systems are not simply augmented versions of the lexical baseline, and are also distinct from each other in their behaviors.[2]

Common characteristics of RTE systems re-

[2]Note that the expected agreement between two random RTE decision-makers is 0.5, so the agreement scores according to Cohen's Kappa measure (Cohen, 1960) are between 0.3 and 0.4.

ported by their designers were the use of structured representations of shallow semantic content (such as augmented dependency parse trees and semantic role labels); the application of NLP resources such as Named Entity recognizers, syntactic and dependency parsers, and coreference resolvers; and the use of special-purpose ad-hoc modules designed to address specific entailment phenomena the researchers had identified, such as the need for numeric reasoning. However, it is not possible to objectively assess the role these capabilities play in each system's performance from the system outputs alone.

2.3 The Need for Detailed Evaluation

An ablation study that formed part of the official RTE 5 evaluation attempted to evaluate the contribution of publicly available knowledge resources such as WordNet (Fellbaum, 1998), VerbOcean (Chklovski and Pantel, 2004), and DIRT (Lin and Pantel, 2001) used by many of the systems. The observed contribution was in most cases limited or non-existent. It is premature, however, to conclude that these resources have little potential impact on RTE system performance: most RTE researchers agree that the real contribution of individual resources is difficult to assess. As the example in figure 1 illustrates, most RTE examples require a number of phenomena to be correctly resolved in order to reliably determine the correct label (the Interaction problem); a perfect coreference resolver might as a result yield little improvement on the standard RTE evaluation, even though coreference resolution is clearly required by human readers in a significant percentage of RTE examples.

Various efforts have been made by individual research teams to address specific capabilities that are intuitively required for good RTE performance, such as (de Marneffe et al., 2008), and the formal treatment of entailment phenomena in (MacCartney and Manning, 2009) depends on and formalizes a divide-and-conquer approach to entailment resolution. But the phenomena-specific capabilities described in these approaches are far from complete, and many are not yet invented. To devote real effort to identify and develop such capabilities, researchers must be confident that the resources (and the will!) exist to create and evaluate their solutions, and that the resource can be shown to be relevant to a sufficiently large subset of the NLP community. While there is widespread belief that there are many relevant entailment phenomena, though each individually may be relevant to relatively few RTE examples (the Sparseness problem), we know of no systematic analysis to determine what those phenomena are, and how sparsely represented they are in existing RTE data.

If it were even known what phenomena were relevant to specific entailment examples, it might be possible to more accurately distinguish system capabilities, and promote adoption of successful solutions to sub-problems. An annotation-side solution also maintains the desirable agnosticism of the RTE problem formulation, by not imposing the requirement on system developers of generating an explanation for each answer. Of course, if examples were also annotated with explanations in a consistent format, this could form the basis of a new evaluation of the kind essayed in the pilot study in (Giampiccolo et al., 2007).

3 Annotation Proposal and Pilot Study

As part of our challenge to the NLP community, we propose a distributed OntoNotes-style approach (Hovy et al., 2006) to this annotation effort: distributed, because it should be undertaken by a diverse range of researchers with interests in different semantic phenomena; and similar to the OntoNotes annotation effort because it should not presuppose a fixed, closed ontology of entailment phenomena, but rather, iteratively hypothesize and refine such an ontology using inter-annotator agreement as a guiding principle. Such an effort would require a steady output of RTE examples to form the underpinning of these annotations; and in order to get sufficient data to represent less common, but nonetheless important, phenomena, a large body of data is ultimately needed.

A research team interested in annotating a new phenomenon should use examples drawn from the common corpus. Aside from any task-specific gold standard annotation they add to the entailment pairs, they should augment existing explanations by indicating in which examples their phenomenon occurs, and at which point in the existing explanation for each example. In fact, this latter effort – identifying phenomena relevant to textual inference, marking relevant RTE examples, and generating explanations – itself enables other researchers to select from known problems, assess their likely impact, and automatically generate rel-

evant corpora.

To assess the feasibility of annotating RTE-oriented local entailment phenomena, we developed an inference model that could be followed by annotators, and conducted a pilot annotation study. We based our initial effort on observations about RTE data we made while participating in RTE challenges, together with intuitive conceptions of the kinds of knowledge that might be available in semi-structured or structured form. In this section, we present our annotation inference model, and the results of our pilot annotation effort.

3.1 Inference Process

To identify and annotate RTE sub-phenomena in RTE examples, we need a defensible model for the entailment process that will lead to consistent annotation by different researchers, and to an extensible framework that can accommodate new phenomena as they are identified.

We modeled the entailment process as one of manipulating the text and hypothesis to be as similar as possible, by first identifying parts of the text that matched parts of the hypothesis, and then identifying connecting structure. Our inherent assumption was that the meanings of the Text and Hypothesis could be represented as sets of n-ary relations, where relations could be connected to other relations (i.e., could take other relations as arguments). As we followed this procedure for a given example, we marked which entailment phenomena were required for the inference. We illustrate the process using the example in figure 1.

First, we would identify the arguments "BMI" and "another company" in the Hypothesis as matching "BMI" and "LexCorp" respectively, requiring 1) *Parent-Sibling* to recognize that "LexCorp" can match "company". We would tag the example as requiring 2) *Nominalization Resolution* to make "purchase" the active relation and 3) *Passivization* to move "BMI" to the subject position. We would then tag it with 4) *Simple Verb Rule* to map "A purchase B" to "A acquire B". These operations make the relevant portion of the Text identical to the Hypothesis, so we are done.

For the same Text, but with Hypothesis 2 (a negative example), we follow the same steps 1-3. We would then use 4) *Lexical Relation* to map "purchase" to "buy". We would then observe that the only possible match for the hypothesis argument "for $3.4Bn" is the text argument "for $2Bn". We would label this as a 5) *Numerical Quantity Mismatch* and 6) *Excluding Argument* (it can't be the case that in the same transaction, the same company was sold for two different prices).

Note that neither explanation mentions the anaphora resolution connecting "they" to "traders", because it is not strictly required to determine the entailment label.

As our example illustrates, this process makes sense for both positive and negative examples. It also reflects common approaches in RTE systems, many of which have explicit alignment components that map parts of the Hypothesis to parts of the Text prior to a final decision stage.

3.2 Annotation Labels

We sought to identify roles for background knowledge in terms of domains and general inference steps, and the types of linguistic phenomena that are involved in representing the same information in different ways, or in detecting key differences in two similar spans of text that indicate a difference in meaning. We annotated examples with domains (such as "Work") for two reasons: to establish whether some phenomena are correlated with particular domains; and to identify domains that are sufficiently well-represented that a knowledge engineering study might be possible.

While we did not generate an explicit representation of our entailment process, i.e. explanations, we tracked which phenomena were strictly required for inference. The annotated corpora and simple CGI scripts for annotation are available at *http://cogcomp.cs.illinois.edu/Data/ACL2010_RTE.php*.

The phenomena that we considered during annotation are presented in Tables 3, 4, 5, and 6. We tried to define each phenomenon so that it would apply to both positive and negative examples, but ran into a problem: often, negative examples can be identified principally by structural differences: the components of the Hypothesis all match components in the Text, but they are not connected by the appropriate structure in the Text. In the case of contradictions, it is often the case that a key relation in the Hypothesis must be matched to an incompatible relation in the Text. We selected names for these structural behaviors, and tagged them when we observed them, but the counterpart for positive examples must always hold: it must necessarily be the case that the structure in the Text linking the arguments that match those in the

Hypothesis must be comparable to the Hypothesis structure. We therefore did not tag this for positive examples.

We selected a subset of 210 examples from the NIST TAC RTE 5 (Bentivogli et al., 2009) Test set drawn equally from the three sub-tasks (IE, IR and QA). Each example was tagged by both annotators. Two passes were made over the data: the first covered 50 examples from each RTE sub-task, while the second covered an additional 20 examples from each sub-task. Between the two passes, concepts the annotators identified as difficult to annotate were discussed and more carefully specified, and several new concepts were introduced based on annotator observations.

Tables 3, 4, 5, and 6 present information about the distribution of the phenomena we tagged, and the inter-annotator agreement (Cohen's Kappa (Cohen, 1960)) for each. "Occurrence" lists the average percentage of examples labeled with a phenomenon by the two annotators.

Domain	Occurrence	Agreement
work	16.90%	0.918
name	12.38%	0.833
die kill injure	12.14%	0.979
group	9.52%	0.794
be in	8.57%	0.888
kinship	7.14%	1.000
create	6.19%	1.000
cause	6.19%	0.854
come from	5.48%	0.879
win compete	3.10%	0.813
Others	29.52%	0.864

Table 3: Occurrence statistics for domains in the annotated data.

Phenomenon	Occurrence	Agreement
Named Entity	91.67%	0.856
locative	17.62%	0.623
Numerical Quantity	14.05%	0.905
temporal	5.48%	0.960
nominalization	4.05%	0.245
implicit relation	1.90%	0.651

Table 4: Occurrence statistics for hypothesis structure features.

From the tables it is apparent that good performance on a range of phenomena in our inference model are likely to have a significant effect on RTE results, with coreference being deemed essential to the inference process for 35% of examples, and a number of other phenomena are sufficiently well represented to merit near-future attention (assuming that RTE systems do not already handle these phenomena, a question we address in section 4). It is also clear from the predominance of *Simple Rewrite Rule* instances, together with the frequency of most of the domains we selected, that knowledge engineering efforts also have a key role in improving RTE performance.

Phenomenon	Occurrence	Agreement
coreference	35.00%	0.698
simple rewrite rule	32.62%	0.580
lexical relation	25.00%	0.738
implicit relation	23.33%	0.633
factoid	15.00%	0.412
parent-sibling	11.67%	0.500
genetive relation	9.29%	0.608
nominalization	8.33%	0.514
event chain	6.67%	0.589
coerced relation	6.43%	0.540
passive-active	5.24%	0.583
numeric reasoning	4.05%	0.847
spatial reasoning	3.57%	0.720

Table 5: Occurrence statistics for entailment phenomena and knowledge resources

Phenomenon	Occurrence	Agreement
missing argument	16.19%	0.763
missing relation	14.76%	0.708
excluding argument	10.48%	0.952
Named Entity mismatch	9.29%	0.921
excluding relation	5.00%	0.870
disconnected relation	4.52%	0.580
missing modifier	3.81%	0.465
disconnected argument	3.33%	0.764
Numeric Quant. mismatch	3.33%	0.882

Table 6: Occurrences of negative-only phenomena

3.3 Discussion

Perhaps surprisingly, given the difficulty of the task, inter-annotator agreement was consistently good to excellent (above 0.6 and 0.8, respectively), with few exceptions, indicating that for most targeted phenomena, the concepts were well-specified. The results confirmed our initial intuition about some phenomena: for example, that coreference resolution is central to RTE, and that detecting the connecting structure is crucial in discerning negative from positive examples. We also found strong evidence that the difference between contradiction and unknown entailment examples is often due to the behavior of certain relations that either preclude certain other relations holding between the same arguments (for example, winning a contest vs. losing a contest), or which can only hold for a single referent in one argument position (for example, "work" relations such as job title are typically constrained so that a single person holds one position).

We found that for some examples, there was more than one way to infer the hypothesis from the text. Typically, for positive examples this involved overlap between phenomena; for example, Coreference might be expected to resolve implicit rela-

tions induced from appositive structures. In such cases we annotated every way we could find.

In future efforts, annotators should record the entailment steps they used to reach their decision. This will make disagreement resolution simpler, and could also form a possible basis for generating gold standard explanations. At a minimum, each inference step must identify the spans of the Text and Hypothesis that are involved and the name of the entailment phenomenon represented; in addition, a partial order over steps must be specified when one inference step requires that another has been completed.

Future annotation efforts should also add a category "Other", to indicate for each example whether the annotator considers the listed entailment phenomena sufficient to identify the label. It might also be useful to assess the difficulty of each example based on the time required by the annotator to determine an explanation, for comparison with RTE system errors.

These, together with specifications that minimize the likely disagreements between different groups of annotators, are processes that must be refined as part of the broad community effort we seek to stimulate.

4 Pilot RTE System Analysis

In this section, we sketch out ways in which the proposed analysis can be applied to learn something about RTE system behavior, even when those systems do not provide anything beyond the output label. We present the analysis in terms of sample questions we hope to answer with such an analysis.

1. If a system needs to improve its performance, which features should it concentrate on? To answer this question, we looked at the top-5 systems and tried to find which phenomena are active in the mistakes they make.

(a) Most systems seem to fail on examples that need *numeric reasoning* to get the entailment decision right. For example, system H got all 10 examples with numeric reasoning wrong.

(b) All top-5 systems make consistent errors in cases where identifying a mismatch in named entities (NE) or numerical quantities (NQ) is important to make the right decision. System G got 69% of cases with NE/NQ mismatches wrong.

(c) Most systems make errors in examples that have a disconnected or exclusion component (argument/relation). System J got 81% of cases with a disconnected component wrong.

(d) Some phenomena are handled well by certain systems, but not by others. For example, failing to recognize a parent-sibling relation between entities/concepts seems to be one of the top-5 phenomena active in systems E and H. System H also fails to correctly label over 53% of the examples having kinship relation.

2. Which phenomena have strong correlations to the entailment labels among hard examples? We called an example hard if at least 4 of the top 5 systems got the example wrong. In our annotation dataset, there were 41 hard examples. Some of the phenomena that strongly correlate with the TE labels on hard examples are: deeper lexical relation between words ($\rho = 0.542$), and need for external knowledge ($\rho = 0.345$). Further, we find that the top-5 systems tend to make mistakes in cases where the lexical approach also makes mistakes ($\rho = 0.355$).

3. What more can be said about individual systems? In order to better understand the system behavior, we wanted to check if we could predict the system behavior based on the phenomena we identified as important in the examples. We learned SVM classifiers over the identified phenomena and the lexical similarity score to predict both the labels and errors systems make for each of the top-5 systems. We could predict all 10 system behaviors with over 70% accuracy, and could predict labels and mistakes made by two of the top-5 systems with over 77% accuracy. This indicates that although the identified phenomena are indicative of the system performance, it is probably too simplistic to assume that system behavior can be easily reproduced solely as a disjunction of phenomena present in the examples.

4. Does identifying the phenomena correctly help learn a better TE system? We tried to learn an entailment classifier over the phenomenon identified and the top 5 system outputs. The results are summarized in Table 7. All reported numbers are 20-fold cross-validation accuracy from an SVM classifier learned over the features mentioned. The results show that correctly identifying the named-entity and numeric quantity mis-

No.	Feature description	No. of feats	Accuracy over which features	
			phenomena	pheno. + sys. labels
(0)	Only system labels	5	—	0.714
(1)	Domain and hypothesis features (Tables 3, 4)	16	0.510	0.705
(2)	(1) + NE + NQ	18	0.619	0.762
(3)	(1) + Knowledge resources (subset of Table 5)	22	0.662	0.762
(4)	(3) + NE + NQ	24	0.738	0.805
(5)	(1) + Entailment and Knowledge resources (Table 5)	29	0.748	0.791
(6)	(5) + negative-only phenomena (Table 6)	38	0.971	0.943

Table 7: Accuracy in predicting the label based on the phenomena and top-5 system labels.

matches improves the overall accuracy significantly. If we further recognize the need for knowledge resources correctly, we can correctly explain the label for 80% of the examples. Adding the entailment and negation features helps us explain the label for 97% of the examples in the annotated corpus.

It must be clarified that the results do not show the textual entailment problem itself is solved with 97% accuracy. However, we believe that if a system could recognize key negation phenomena such as *Named Entity mismatch*, presence of *Excluding arguments*, etc. correctly and consistently, it could model them as a *Contradiction* features in the final inference process to significantly improve its overall accuracy. Similarly, identifying and resolving the key entailment phenomena in the examples, would boost the inference process in positive examples. However, significant effort is still required to obtain near-accurate knowledge and linguistic resources.

5 Discussion

NLP researchers in the broader community continually seek new problems to solve, and pose more ambitious tasks to develop NLP and NLU capabilities, yet recognize that even solutions to problems which are considered "solved" may not perform as well on domains different from the resources used to train and develop them. Solutions to such NLP tasks could benefit from evaluation and further development on corpora drawn from a range of domains, like those used in RTE evaluations.

It is also worthwhile to consider each task as part of a larger inference process, and therefore motivated not just by performance statistics on special-purpose corpora, but as part of an interconnected web of resources; and the task of Recognizing Textual Entailment has been designed to exercise a wide range of linguistic and reasoning capabilities.

The entailment setting introduces a potentially broader context to resource development and assessment, as the hypothesis and text provide context for each other in a way different than local context from, say, the same paragraph in a document: in RTE's positive examples, the Hypothesis either restates some part of the Text, or makes statements inferable from the statements in the Text. This is not generally true of neighboring sentences in a document. This distinction opens the door to "purposeful", or goal-directed, inference in a way that may not be relevant to a task studied in isolation.

The RTE community seems mainly convinced that incremental advances in local entailment phenomena (including application of world knowledge) are needed to make significant progress. They need ways to identify sub-problems of textual inference, and to evaluate those solutions both in isolation and in the context of RTE. RTE system developers are likely to reward well-engineered solutions by adopting them and citing their authors, because such solutions are easier to incorporate into RTE systems. They are also more likely to adopt solutions with established performance levels. These characteristics promote publication of software developed to solve NLP tasks, attention to its usability, and publication of materials supporting reproduction of results presented in technical papers.

For these reasons, we assert that RTE is a natural motivator of new NLP tasks, as researchers look for components capable of improving performance; and that RTE is a natural setting for evaluating solutions to a broad range of NLP problems, though not in its present formulation: we must solve the problem of credit assignment, to recognize component contributions. We have therefore proposed a suitable annotation effort, to provide the resources necessary for more detailed evaluation of RTE systems.

We have presented a linguistically-motivated

analysis of entailment data based on a step-wise procedure to resolve entailment decisions, intended to allow independent annotators to reach consistent decisions, and conducted a pilot annotation effort to assess the feasibility of such a task.

We do not claim that our set of domains or phenomena are complete: for example, our illustrative example could be tagged with a domain *Mergers and Acquisitions*, and a different team of researchers might consider *Nominalization Resolution* to be a subset of *Simple Verb Rules*. This kind of disagreement in coverage is inevitable, but we believe that in many cases it suffices to introduce a new domain or phenomenon, and indicate its relation (if any) to existing domains or phenomena. In the case of introducing a non-overlapping category, no additional information is needed. In other cases, the annotators can simply indicate the phenomena being merged or split (or even replaced). This information will allow other researchers to integrate different annotation sources and maintain a consistent set of annotations.

6 Conclusions

In this paper, we have presented a case for a broad, long-term effort by the NLP community to coordinate annotation efforts around RTE corpora, and to evaluate solutions to NLP tasks relating to textual inference in the context of RTE. We have identified limitations in the existing RTE evaluation scheme, proposed a more detailed evaluation to address these limitations, and sketched a process for generating this annotation. We have proposed an initial annotation scheme to prompt discussion, and through a pilot study, demonstrated that such annotation is both feasible and useful.

We ask that researchers not only contribute task specific annotation to the general pool, and indicate how their task relates to those already added to the annotated RTE corpora, but also invest the additional effort required to augment the cross-domain annotation: marking the examples in which their phenomenon occurs, and augmenting the annotator-generated explanations with the relevant inference steps.

These efforts will allow a more meaningful evaluation of RTE systems, and of the component NLP technologies they depend on. We see the potential for great synergy between different NLP subfields, and believe that all parties stand to gain from this collaborative effort. We therefore respectfully suggest that you "ask not what RTE can do for you, but what you can do for RTE..."

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This research was partly sponsored by Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181, by a grant from Boeing and by MIAS, the Multimodal Information Access and Synthesis center at UIUC, part of CCICADA, a DHS Center of Excellence. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the sponsors.

References

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth pascal recognizing textual entailment challenge. In *Notebook papers and Results, Text Analysis Conference (TAC)*, pages 14–24.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 33–40.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

I. Dagan, O. Glickman, and B. Magnini, editors. 2006. *The PASCAL Recognising Textual Entailment Challenge.*, volume 3944. Springer-Verlag, Berlin.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047, Columbus, Ohio, June. Association for Computational Linguistics.

Quang Do, Dan Roth, Mark Sammons, Yuancheng Tu, and V.G.Vinod Vydiswaran. 2010. Robust, Light-weight Approaches to compute Lexical Similarity. Computer Science Research and Technical Reports, University of Illinois. http://L2R.cs.uiuc.edu/∼danr/Papers/DRSTV10.pdf.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, June. Association for Computational Linguistics.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 905–912, Sydney, Australia, July. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of HLT/NAACL*, New York.

D. Lin and P. Pantel. 2001. DIRT: discovery of inference rules from text. In *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pages 323–328.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *The Eighth International Conference on Computational Semantics (IWCS-8)*, Tilburg, Netherlands.

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *ACL/AFNLP*, pages 791–799, Suntec, Singapore, August. Association for Computational Linguistics.

Sebastian Pado, Michel Galley, Dan Jurafsky, and Christopher D. Manning. 2009. Robust machine translation evaluation with entailment features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 297–305, Suntec, Singapore, August. Association for Computational Linguistics.

Assessing the Role of Discourse References in Entailment Inference

Shachar Mirkin, Ido Dagan
Bar-Ilan University
Ramat-Gan, Israel
{mirkins,dagan}@cs.biu.ac.il

Sebastian Padó
University of Stuttgart
Stuttgart, Germany
pado@ims.uni-stuttgart.de

Abstract

Discourse references, notably coreference and bridging, play an important role in many text understanding applications, but their impact on textual entailment is yet to be systematically understood. On the basis of an in-depth analysis of entailment instances, we argue that discourse references have the potential of substantially improving textual entailment recognition, and identify a number of research directions towards this goal.

1 Introduction

The detection and resolution of *discourse references* such as coreference and bridging anaphora play an important role in text understanding applications, like question answering and information extraction. There, reference resolution is used for the purpose of combining knowledge from multiple sentences. Such knowledge is also important for Textual Entailment (TE), a generic framework for modeling semantic inference. TE reduces the inference requirements of many text understanding applications to the problem of determining whether the meaning of a given textual assertion, termed *hypothesis* (H), can be inferred from the meaning of certain *text* (T) (Dagan et al., 2006).

Consider the following example:

(1) T: "*Not only had he developed an aversion to the* ***President***$_1$ *and politics in general,* ***Oswald***$_2$ *was also a failure with Marina, his wife. [...] Their relationship was supposedly responsible for why* ***he***$_2$ *killed* ***Kennedy***$_1$."

H: "*Oswald killed President Kennedy.*"

The understanding that the second sentence of the text entails the hypothesis draws on two coreference relationships, namely that *he* is *Oswald*, and that the *Kennedy* in question is *President Kennedy*. However, the utilization of discourse information for such inferences has been so far limited mainly to the substitution of nominal coreferents, while many aspects of the interface between discourse and semantic inference needs remain unexplored.

The recently held Fifth Recognizing Textual Entailment (RTE-5) challenge (Bentivogli et al., 2009a) has introduced a *Search* task, where the text sentences are interpreted in the context of their full discourse, as in Example 1 above. Accordingly, TE constitutes an interesting framework – and the Search task an adequate dataset – to study the interrelation between discourse and inference.

The goal of this study is to analyze the roles of discourse references for textual entailment inference, to provide relevant findings and insights to developers of both reference resolvers and entailment systems and to highlight promising directions for the better incorporation of discourse phenomena into inference. Our focus is on a manual, in-depth assessment that results in a classification and quantification of discourse reference phenomena and their utilization for inference. On this basis, we develop an account of formal devices for incorporating discourse references into the inference computation. An additional point of interest is the interrelation between entailment knowledge and coreference. E.g., in Example 1 above, knowing that Kennedy was a president can alleviate the need for coreference resolution. Conversely, coreference resolution can often be used to overcome gaps in entailment knowledge.

Structure of the paper. In Section 2, we provide background on the use of discourse references in natural language processing (NLP) in general and specifically in TE. Section 3 describes the goals of this study, followed by our analysis scheme (Section 4) and the required inference

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1209–1219,
Uppsala, Sweden, 11-16 July 2010.

mechanisms (Section 5). Section 6 presents quantitative findings and further observations. Conclusions are discussed in Section 7.

2 Background

2.1 Discourse in NLP

Discourse information plays a role in a range of NLP tasks. It is obviously central to *discourse processing* tasks such as text segmentation (Hearst, 1997). Reference information provided by discourse is also useful for *text understanding* tasks such as question answering (QA), information extraction (IE) and information retrieval (IR) (Vicedo and Ferrndez, 2006; Zelenko et al., 2004; Na and Ng, 2009), as well as for the acquisition of lexical-semantic "narrative schema" knowledge (Chambers and Jurafsky, 2009). Discourse references have been the subject of attention in both the Message Understanding Conference (Grishman and Sundheim, 1996) and the Automatic Content Extraction program (Strassel et al., 2008).

The simplest form of information that discourse provides is *coreference*, i.e., information that two linguistic expressions refer to the same entity or event. Coreference is particularly important for processing pronouns and other anaphoric expressions, such as *he* in Example 1. Ability to resolve this reference translates directly into, e.g., a QA system's ability to answer questions like *Who killed Kennedy?*.

A second, more complex type of information stems from *bridging references*, such as in the following discourse (Asher and Lascarides, 1998):

(2) "*I've just **arrived**. The **camel** is outside.*"

While coreference indicates equivalence, bridging points to the existence of a salient semantic relation between two distinct entities or events. Here, it is (informally) *'means of transport'*, which would make the discourse (2) relevant for a question like *How did I arrive here?*. Other types of bridging relations include set-membership, roles in events and consequence (Clark, 1975).

Note, however, that text understanding systems are generally limited to the resolution of entity (or even just pronoun) coreference, e.g. (Li et al., 2009; Dali et al., 2009). An important reason is the unavailability of tools to resolve the more complex (and difficult) forms of discourse reference such as event coreference and bridging.[1] Another reason is uncertainty about their practical importance.

2.2 Discourse in Textual Entailment

Textual Entailment has been introduced in Section 1 as a common-sense notion of inference. It has spawned interest in the computational linguistics community as a common denominator of many NLP tasks including IE, summarization and tutoring (Romano et al., 2006; Harabagiu et al., 2007; Nielsen et al., 2009).

Architectures for Textual Entailment. Over the course of recent RTE challenges (Giampiccolo et al., 2007; Giampiccolo et al., 2008), the main benchmark for TE technology, two architectures for modeling TE have emerged as dominant: *transformations* and *alignment*. The goal of ***transformation***-based TE models is to determine the entailment relation $T \Rightarrow H$ by finding a "proof", i.e., a sequence of consequents, $(T, T_1, \ldots, T_n)$, such that T_n=H (Bar-Haim et al., 2008; Harmeling, 2009), and that in each transformation, $T_i \rightarrow T_{i+1}$, the consequent T_{i+1} is entailed by T_i. These transformations commonly include lexical modifications and the generation of syntactic alternatives. The second major approach constructs an ***alignment*** between the linguistic entities of the trees (or graphs) of T and H, which can represent syntactic structure, semantic structure, or non-hierarchical phrases (Zanzotto et al., 2009; Burchardt et al., 2009; MacCartney et al., 2008). H is assumed to be entailed by T if its entities are aligned "well" to corresponding entities in T. Alignment quality is generally determined based on features that assess the validity of the local replacement of the T entity by the H entity.

While transformation- and alignment-based entailment models look different at first glance, they ultimately have the same goal, namely obtaining a maximal *coverage* of H by T, i.e. to identify matches of as many elements of H within T as possible.[2] To do so, both architectures typically make use of *inference rules* such as '*Y was purchased by X* $\rightarrow$ *X paid for Y*', either by directly applying them as transformations, or by using them

[1] Some studies, e.g. (Markert et al., 2003; Poesio et al., 2004), address the resolution of a few specific kinds of bridging relations; yet, wide-scope systems for bridging resolution are unavailable.

[2] Clearly, the details of how the final entailment decision is made based on the attained coverage differ substantially among models.

to score alignments. Rules are generally drawn from external knowledge resources, such as WordNet (Fellbaum, 1998) or DIRT (Lin and Pantel, 2001), although knowledge gaps remain a key obstacle (Bos, 2005; Balahur et al., 2008; Bar-Haim et al., 2008).

Discourse in previous RTE challenges. The first two rounds of the RTE challenge used "self-contained" texts and hypotheses, where discourse considerations played virtually no role. A first step towards a more comprehensive notion of entailment was taken with RTE-3 (Giampiccolo et al., 2007), when paragraph-length texts were first included and constituted 17% of the texts in the test set. Chambers et al. (2007) report that in a sample of $T - H$ pairs drawn from the development set, 25% involved discourse references.

Using the concepts introduced above, the impact of discourse references can be generally described as a *coverage problem*, independent of the system's architecture. In Example 1, the hypothesis word *Oswald* cannot be safely linked to the text pronoun *he* without further knowledge about *he*; the same is true for '*Kennedy* $\rightarrow$ *President Kennedy*' which involves a specialization that is only warranted in the specific discourse.

A number of systems have tried to address the question of coreference in RTE as a preprocessing step prior to inference proper, with most systems using off-the-shelf coreference resolvers such as JavaRap (Qiu et al., 2004) or OpenNLP[3]. Generally, anaphoric expressions were textually replaced by their antecedents. Results were inconclusive, however, with several reports about errors introduced by automatic coreference resolution (Agichtein et al., 2008; Adams et al., 2007). Specific evaluations of the contribution of coreference resolution yielded both small negative (Bar-Haim et al., 2008) and insignificant positive (Chambers et al., 2007) results.

3 Motivation and Goals

The results of recent studies, as reported in Section 2.2, seem to show that current resolution of discourse references in RTE systems hardly affects performance. However, our intuition is that these results can be attributed to four major limitations shared by these studies: (1) the datasets, where discourse phenomena were not well represented; (2) the off-the-shelf coreference resolution systems which may have been not robust enough; (3) the limitation to nominal coreference; and (4) overly simple integration of reference information into the inference engines.

The goal of this paper is to assess the impact of discourse references on entailment with an annotation study which removes these limitations. To counteract (1), we use the recent RTE-5 Search dataset (details below). To avoid (2), we perform a manual analysis, assuming discourse references as predicted by an oracle. With regards to (3), our annotation scheme covers coreference and bridging relations of all syntactic categories and classifies them. As for (4), we suggest several operations necessary to integrate the discourse information into an entailment engine.

In contrast to the numerous existing datasets annotated for discourse references (Hovy et al., 2006; Strassel et al., 2008), we do not annotate exhaustively. Rather, we are interested specifically in those references instances that impact inference. Furthermore, we analyze each instance from an entailment perspective, characterizing the relevant factors that have an impact on inference. To our knowledge, this is the first such in-depth study.[4]

The results of our study are of twofold interest. First, they provide guidance for the developers of reference resolvers who might prioritize the scope of their systems to make them more valuable for inference. Second, they point out potential directions for the developers of inference systems by specifying what additional inference mechanisms are needed to utilize discourse information.

The RTE-5 Search dataset. We base our annotation on the Search task dataset, a new addition to the recent Fifth RTE challenge (Bentivogli et al., 2009a) that is motivated by the needs of NLP applications and drawn from the TAC summarization track. In the Search task, TE systems are required to find *all* individual sentences in a given corpus which entail the hypothesis – a setting that is sensible not only for summarization, but also for information access tasks like QA. Sentences are judged individually, but "are to be interpreted in the context of the corpus as they rely on explicit and implicit references to entities, events, dates, places, etc., mentioned elsewhere in the corpus" (Bentivogli et al., 2009b).

[3] http://opennlp.sourceforge.net

[4] The guidelines and the dataset are available at http://www.cs.biu.ac.il/~nlp/downloads/

	Text		Hypothesis
i	T'	*Once the reform becomes law, Spain will join the Netherlands and Belgium in allowing* ***homosexual marriages.***	*Massachusetts allows* ***homosexual marriages***
	T	***Such unions*** *are also legal in six Canadian provinces and the northeastern US state of Massachusetts.*	
ii	T'	*The official name of* ***2003 UB313*** *has yet to be determined.*	***2003 UB313*** *is in the Kuiper Belt*
	T	*Brown said he expected to find a moon orbiting* ***Xena*** *because many Kuiper Belt objects are paired with moons.*	
iii	T'_a	*All seven aboard the* ***AS-28 submarine*** *appeared to be in satisfactory condition, naval spokesman said.*	*The* ***AS-28 mini submarine*** *was trapped underwater*
	T'_b	*British crews were working with Russian naval authorities to maneuver the unmanned robotic vehicle and untangle the* ***AS-28****.*	
	T	*The Russian military was racing against time early Friday to rescue a* ***mini submarine trapped on the seabed.***	
iv	T'	***China*** *seeks solutions to its coal mine safety.*	*A* ***mining accident in China*** *has killed several miners*
	T	*A* ***recent accident*** *has cost more than a dozen miners their lives.*	
v	T''	*A remote-controlled device was lowered to the stricken vessel to cut the cables in which the* ***AS-28 vehicle*** *is caught.*	*The* ***AS-28 mini submarine*** *was trapped underwater*
	T'	*The* ***mini submarine*** *was resting on the seabed at a depth of about 200 meters.*	
	T	*Specialists said* ***it*** *could have become tangled up with a metal cable or in sunken nets from a fishing trawler.*	
vi	T	*. . . dried up* ***lakes in Siberia****, because the permafrost beneath* ***them*** *has begun to thaw.*	*The ice is* ***melting in the Arctic***

Table 1: Examples for discourse-dependent entailment in the RTE-5 dataset, where the inference of H depends on reference information from the discourse sentences T' / T''. Referring terms (in T) and target terms (in H) are shown in boldface.

4 Analysis Scheme

For annotating the RTE-5 data, we operationalize reference relations that are *relevant* for entailment as those that improve *coverage*. Recall from Section 2.2 that the concept of coverage is applicable to both transformation and alignment models, all of which aim at maximizing coverage of H by T.

We represent T and H as syntactic trees, as common in the RTE literature (Zanzotto et al., 2009; Agichtein et al., 2008). Specifically, we assume MINIPAR-style (Lin, 1993) dependency trees where nodes represent text expressions and edges represent the syntactic relations between them. We use "term" to refer to text expressions, and "components" to refer to nodes, edges, and subtrees. Dependency trees are a popular choice in RTE since they offer a fairly semantics-oriented account of the sentence structure that can still be constructed robustly. In an ideal case of entailment, all nodes *and* dependency edges of H are covered by T.

For each $T - H$ pair, we annotate all relevant discourse references in terms of three items: the *target component* in H, the *focus term* in T, and the *reference term* which stands in a reference relation to the focus term. By resolving this reference, the target component can usually be inferred; sometimes, however, more than one reference term needs to be found. We now define and illustrate these concepts on examples from Table 1.[5]

The ***target component*** is a tree component in H that cannot be covered by the "local" material from T. An example for a tree component is Example (v), where the target component *AS-28 mini submarine* in H cannot be inferred from the pronoun *it* in T. Example (vi) demonstrates an edge as target component. In this case, the edge in H connecting *melt* with the modifier *in the Arctic* is not found in T. Although each of the hypothesis' nodes can be covered separately via knowledge-based rules (e.g. '*Siberia* → *Arctic*', '*permafrost* → *ice*', '*thaw* ↔ *melt*'), the resulting fragments in T are unconnected without the (intra-sentential) coreference between *them* and *lakes in Siberia*.

For each target component, we identify its ***focus term*** as the expression in T that does not cover the target component itself but participates in a reference relation that can help covering it.

We follow the focus term's reference chain to a ***reference term*** which can, either separately or in combination with the focus term, help covering the target component. In Example (ii), where the

[5] In our annotation, we assume throughout that some knowledge about basic admissible transformations is available, such as passive to active or derivational transformations; for brevity, we ignore articles in the examples and treat named entities as single nodes.

target component in H is *2003 UB313*, *Xena* is the focus term in T and the reference term is a mention of *2003 UB313* in a previous sentence, T'. In this case, the reference term covers the entire target component on its own.

An additional attribute that we record for each instance is whether resolving the discourse reference is *mandatory* for determining entailment, or *optional*. In Example (v), it is mandatory: the inference cannot be completed without the knowledge provided by the discourse. In contrast, in Example (ii), inferring *2003 UB313* from *Xena* is optional. It can be done either by identifying their coreference relation, or by using background knowledge in the form of an entailment rule, '*Xena* $\leftrightarrow$ *2003 UB313*', that is applicable in the context of astronomy. Optional discourse references represent instances where discourse information and TE knowledge are interchangeable. As mentioned, knowledge gaps constitute a major obstacle for TE systems, and we cannot rely on the availability of any ceratin piece of knowledge to the inference process. Thus, in our scheme, mandatory references provide a "lower bound" with regards to the necessity to resolve discourse references, even in the presence of *complete knowledge*; optional references, on the other hand, set an "upper bound" for the contribution of discourse resolution to inference, when *no knowledge* is available. At the same time, this scheme allows investigating how much TE knowledge can be replaced by (perfect) discourse processing.

When choosing a reference term, we search the reference chain of the focus term for the nearest expression that is identical to the target component or a subcomponent of it. If we find such an expression, covering the identical part of the target component requires no entailment knowledge. If no identical reference term exists, we choose the semantically 'closest' term from the reference chain, i.e. the term which requires the least knowledge to infer the target component. For instance, we may pick *permafrost* as the semantically closet term to the target *ice* if the latter is not found in the focus term's reference chain.

Finally, for each reference relation that we annotate, we record four additional attributes which we assumed to be informative in an evaluation. First, the *reference type*: Is the relation a coreference or a bridging reference? Second, the *syntactic type* of the focus and reference terms. Third, the *focus/reference terms entailment status* – does some kind of entailment relation hold between the two terms? Fourth, the *operation* that should be performed on the focus and reference terms to obtain coverage of the target component (as specified in Section 5).

5 Integrating Discourse References into Entailment Recognition

In initial analysis we found that the standard substitution operation applied by virtually all previous studies for integrating coreference into entailment is insufficient. We identified three distinct cases for the integration of discourse reference knowledge in entailment, which correspond to different relations between the target component, the focus term and the reference term. This section describes the three cases and characterizes them in terms of tree transformations. An initial version of these transformations is described in (Abad et al., 2010). We assume a transformation-based entailment architecture (cf. Section 2.2), although we believe that the key points of our account are also applicable to alignment-based architecture. Transformations create revised trees that cover previously uncovered target components in H. The output of each transformation, T_1, is comprised of copies of the components used to construct it, and is appended to the discourse forest, which includes the dependency trees of all sentences and their generated consequents.

We assume that we have access to a dependency tree for H, a dependency forest for T and its discourse context, as well as the output of a perfect discourse processor, i.e., a complete set of both coreference and bridging relations, including the type of bridging relation (e.g. *part-of*, *cause*).

We use the following notation. We use x, y for tree nodes, and S_x to denote a (sub-)tree with root x. $lab(x)$ is the label of the incoming edge of x (i.e., its grammatical function). We write $C(x, y)$ for a coreference relation between S_x and S_y, the corresponding trees of the focus and reference terms, respectively. We write $B_r(x, y)$ for a bridging relation, where r is its type.

(1) Substitution: This is the most intuitive and widely-used transformation, corresponding to the treatment of discourse information in existing systems. It applies to coreference relations, when an expression found elsewhere in the text (the reference term) can cover all missing information (the

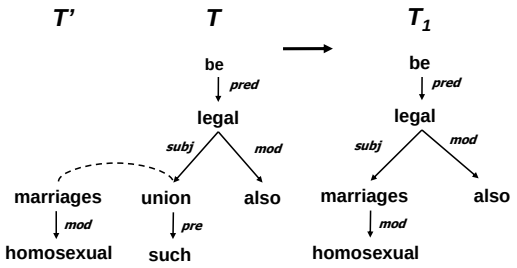


Figure 1: The *Substitution* transformation, demonstrated on the relevant subtrees of Example (i). The dashed line denotes a discourse reference.

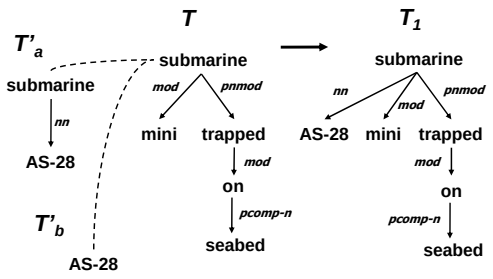


Figure 2: The *dependent-merge* (T'_a) and *head-merge* (T'_b) transformations (Example (iii)).

target component) on its own. In such cases, the reference term can replace the entire focus term. Apparently (cf. Section 6), substitution applies also to some types of bridging relations, such as *set-membership*, when the member is sufficient for representing the entire set for the necessary inference. For example, in "*I met two people yesterday. The woman told me a story.*" (Clark, 1975), substituting *two people* with *woman* results in a text which is entailed from the discourse, and which allows inferring "*I met a woman yesterday.*"

In a parse tree representation, given a coreference relation $C(x, y)$ (or $B_r(x, y)$), the newly generated tree, T_1, consists of a copy of T, where the entire tree S_x is replaced by a copy of S_y . In Figure 1, which shows Example (i) from Table 1, *such unions* is substituted by *homosexual marriages*.

Head-substitution. Occasionally, substituting only the head of the focus term is sufficient. In such cases, only the root nodes x and y are substituted. This is the case, for example, with synonymous verbs with identical subcategorization frames (like *melt* and *thaw*). As verbs typically constitute tree roots in dependency parses, substituting or merging (see below) their entire trees might be inappropriate or wasteful. In such cases, the simpler head-substitution may be applied.

(2) Merge: In contrast to substitution, where a match for the entire target component is found elsewhere in the text, this transformation is required when parts of the missing information are scattered among multiple locations in the text. We distinguish between two types of merge transformations: (a) *dependent-merge*, and (b) *head-merge*, depending on the syntactic roles of the merged components.

(a) Dependent-Merge. This operation is applicable when the head of either the focus or reference terms (of both) matches the head node of the target component, but modifiers from both of them are required to cover the target component's dependents. The modifiers are therefore merged as dependents of a single head node, to create a tree that covers the entire target component. Dependent-merge is illustrated in Figure 2, using Example (iii). The component we wish to cover in H is the noun phrase *AS-28 mini submarine*. Unfortunately, the focus term in T, "*mini submarine trapped on the seabed*", covers only the modifier *mini*, but not *AS-28*. This modifier can however be provided by the coreferent term in T'_a (left upper corner). Once merged, the inference engine can, e.g., employ the rule '*on seabed* → *underwater*' to cover H completely.

Formally, assume without loss of generality that y, the reference term's head, matches the root node of the target component. Given $C(x, y)$, we define T_1 as a copy of T, where (i) the subtree S_x is replaced by S_y, and (ii) for all children c of x, a copy of S_c is placed under the copy of y in T_1 with its original edge label, $lab(c)$.

(b) Head-merge. An alternative way to recover the missing information in Example (iii) is to find a reference term whose head word itself (rather than one of its modifiers) matches the target component's missing dependent, as with *AS-28* in Figure 2 in the bottom left corner (T'_b). In terms of parse trees, we need to add one tree as a dependent of the other. Formally, given $C(x, y)$, similarly to dependent-merge, T_1 is created as a copy of T where the subtree S_x is replaced by either S_x or S_y, depending on whichever of x and y matches the target component's head. Assume it is x, for example. Then, a copy of S_y is added as a new child to x. In our sample, head-merge operations correspond to internal coreferences within nominal target components (such as between *AS-28* and *mini submarine* in this case). The appropriate label, $lab(y)$, in these cases is nn (nominal modi-

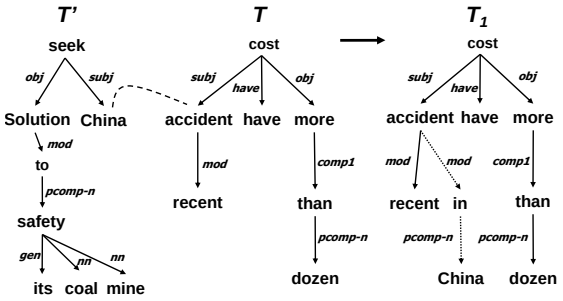


Figure 3: The *insertion* transformation. Dotted edges mark the newly inserted path (Ex. (iv)).

fier). Further analysis is required to specify what other dependencies can hold between such coreferring heads.

(3) Insertion: The last transformation, insertion, is used when a relation that is realized in H is missing from T and is only implied via a bridging relation. In Example (iv), the location that is explicitly mentioned in H can only be covered by T by resolving a bridging reference with *China* in T'. To connect the bridging referents, a new tree component representing the bridging relation is inserted into the consequent tree T_1. In this example, the component connects *China* and *recent accident* via the *in* preposition. Formally, given a bridging relation $B_r(x, y)$, we introduce a new subtree S^r_z into T_1, where z is a child of x and $lab(z) = lab_r$. S^r_z must contain a variable node that is instantiated with a copy of $S(y)$.

This transformation stands out from the others in that it introduces new material. For each bridging relation, it adds a specific subtrees S^r via an edge labeled with lab_r. These two items form the dependency representation of the bridging relation B_r and must be provided by the interface between the discourse and the inference systems. Clearly, their exact form depends on the set of bridging relations provided by the discourse resolver as well as the details of the dependency parses.

As shown in Figure 3, the bridging relation *located-in* (r) is represented by inserting a subtree S^r_z headed by *in* (z) into T_1 and connecting it to *accident* (x) as a modifier (lab_r). The subtree S^r_z consists of a variable node which is connected to *in* with a *pcomp-n* dependency (a nominal head of a prepositional phrase), and which is instantiated with the node *China* (y) when the transformation is applied. Note that the structure of S^r_z and the way it is inserted into T_1 are predefined by the abovementioned interface; only the node to which it is attached and the contents of the variable node are determined at transformation-time.

As another example, consider the following short text from (Clark, 1975): *John was murdered yesterday. The knife lay nearby*. Here, the bridging relation between the murder event and the instrument, *the knife* (x), can be addressed by inserting under x a subtree for the clause *with which* as S^r_z, with a variable which is instantiated by the parse-tree (headed by *murdered*, y) of the entire first sentence *John was murdered yesterday*.

Transformation chaining. Since our transformations are defined to be minimal, some cases require the application of multiple transformations to achieve coverage. Consider Example (v), Table 1. We wish to cover *AS-28 mini submarine* in H from the coreferring *it* in T, *mini submarine* in T' and *AS-28 vehicle* in T''. A substitution of *it* by either coreference does not suffice, since none of the antecedents contains all necessary modifiers. It is therefore necessary to substitute *it* first by one of the coreferences and then merge it with the other.

6 Results

We analyzed 120 sentence-hypothesis pairs of the RTE-5 development set (21 different hypotheses, 111 distinct sentences, 53 different documents). Below, we summarize our findings, focusing on the relation between our findings and the assumptions of previous studies as discussed in Section 3.

General statistics. We found that 44% of the pairs contained reference relations whose resolution was mandatory for inference. In another 28%, references could optionally support the inference of the hypothesis. In the remaining 28%, references did not contribute towards inference. The total number of relevant references was 137, and 37 pairs (27%) contained multiple relevant references. These numbers support our assumption that discourse references play an important role in inference.

Reference types. 73% of the identified references are coreferences and 27% are bridging relations. The most common bridging relation was the location of events (e.g. *Arctic* in ice melting events), generally assumed to be known throughout the document. Other bridging relations we encountered include cause (e.g. between *injured* and *attack*), event participants and set membership.

(%)	Pronoun	NE	NP	VP
Focus term	9	19	49	23
Reference term	-	43	43	14

Table 2: Syntactic types of discourse references

(%)	Sub.	Merge	Insertion
Coreference	62	38	-
Bridging	30	-	70
Total	54	28	18

Table 3: Distribution of transformation types

Syntactic types. Table 2 shows that 77% of all focus terms and 86% of the reference terms were nominal phrases, which justifies their prominent position in work on anaphora and coreference resolution. However, almost a quarter of the focus terms were verbal phrases. We found these focus terms to be frequently crucial for entailment since they included the main predicate of the hypothesis.[6] This calls for an increased focus on the resolution of event references.

Transformations. Table 3 shows the relative frequencies of all transformations. Again, we found that the "default" transformation, substitution, is the most frequent one, and is helpful for both coreference and bridging relations. Substitution is particularly useful for handling pronouns (14% of all substitution instances), the replacement of named entities by synonymous names (32%), the replacement of other NPs (38%), and the substitution of verbal head nodes in event coreference (16%). Yet, in nearly half the cases, a different transformation had to be applied. Insertion accounts for the majority of bridging cases. Head-merge is necessary to integrate proper nouns as modifiers of other head nouns. Dependent-merge, responsible for 85% of the merge transformations, can be used to complete nominal focus terms with missing modifiers (e.g., adjectives), as well as for merging other dependencies between coreferring predicates. This result indicates the importance of incorporating other transformations into inference systems.

Distance of reference terms. The distance between the focus and the reference terms varied considerably, ranging from intra-sentential reference relations and up to several dozen sentences. For more than a quarter of the focus terms, we had to go to other documents to find reference terms that, possibly in conjunction with the focus term, could cover the target components. Interestingly, all such cases involved coreference (about equally divided between the merge transformations and substitutions), while bridging was always "document-local". This result reaffirms the usefulness of cross-document coreference resolution for inference (Huang et al., 2009).

Discourse resolution as preprocessing? In existing RTE systems, discourse references are typically resolved as a preprocessing step. While our annotation was manual and cannot yield direct results about processing considerations, we observed that discourse relations often hold between complex, and deeply embedded, expressions, which makes their automatic resolution difficult. Of course, many RTE systems attempt to normalize and simplify H and T, e.g., by splitting conjunctions or removing irrelevant clauses, but these operations are usually considered a part of the inference rather the preprocessing phase (cf. e.g., Bar-Haim et al. (2007)). Since the resolution of discourse references is likely to profit from these steps, it seems desirable to "postpone" it until after simplification. In transformation-based systems, it might be natural to add discourse-based transformations to the set of inference operations, while in alignment-based systems, discourse references can be integrated into the computation of alignment scores.

Discourse references vs. entailment knowledge. We have stated before that even if a discourse reference is not strictly necessary for entailment, it may be interesting because it represents an alternative to the use of knowledge rules to cover the hypothesis. Sometimes, these rules are generally applicable (e.g., '*Alaska* → *Arctic*'). However, often they are context-specific. Consider the following sentence as T for the hypothesis H: "*The ice is melting in the Arctic*":

(3) T: "*The scene at the* ***receding*** *edge of the Exit Glacier was part festive gathering, part nature tour with an apocalyptic edge.*"

While it is possible to cover *melting* using a rule '*melting* ↔ *receding*', this rule is only valid under quite specific conditions (e.g., for the subject *ice*). Instead of determining the applicability of the rule, a discourse-aware system can take the next sen-

[6] The lower proportion of VPs among reference terms stems from bridging relations between VPs and nominal dependents, such as the abovementioned "location" relation.

tence into account, which contains a coreferring event to *receding* that can cover *melting* in H:

(4) T': "*... people moved closer to the rope line near the glacier as it shied away, practically groaning and* ***melting*** *before their eyes.*"

Discourse relations can in fact encode arbitrarily complex world knowledge, as in the following pair:

(5) H: "*The* ***serial*** *killer BTK was accused of at least 7 killings starting in the 1970's.*"

T: "*Police say BTK may have killed as many as 10 people between 1974 and 1991.*"

Here, the H modifier *serial*, which does not occur in T, can be covered either by world knowledge (a person who killed 10 people is a serial killer), or by resolving the coreference of *BTK* to the term *the serial killer BTK* which occurs in the discourse around T. Our conclusion is that not only can discourse references often replace world knowledge in principle, in practice it often seems easier to resolve discourse references than to determine whether a rule is applicable in a given context or to formalize complex world knowledge as inference rules. Our annotation provides further empirical support to this claim: An entailment relation exists between the focus and reference terms in 60% of the focus-reference term pairs, and in many of the remainder, entailment holds between the terms' heads. Thus, discourse provides relations which are many times equivalent to entailment knowledge rules and can therefore be utilized in their stead.

7 Conclusions

This work has presented an analysis of the relation between discourse references and textual entailment. We have identified a set of limitations common to the handling of discourse relations in virtually all entailment systems. They include the use of off-the-shelf resolvers that concentrate on nominal coreference, the integration of reference information through substitution, and the RTE evaluation schemes, which played down the role of discourse. Since in practical settings, discourse plays an important role, our goal was to develop an agenda for improving the handling of discourse references in entailment-based inference.

Our manual analysis of the RTE-5 dataset shows that while the majority of discourse references that affect inference are nominal coreference relations, another substantial part is made up by verbal terms and bridging relations. Furthermore, we have demonstrated that substitution alone is insufficient to extract all relevant information from the wide range of discourse references that are frequently relevant for inference. We identified three general cases, and suggested matching operations to obtain the relevant inferences, formulated as tree transformations. Furthermore, our evidence suggests that for practical reasons, the resolution of discourse references should be tightly integrated into entailment systems instead of treating it as a preprocessing step.

A particularly interesting result concerns the interplay between discourse references and entailment knowledge. While semantic knowledge (e.g., from WordNet or Wikipedia) has been used beneficially for coreference resolution (Soon et al., 2001; Ponzetto and Strube, 2006), reference resolution has, to our knowledge, not yet been employed to validate entailment rules' applicability. Our analyses suggest that in the context of deciding textual entailment, reference resolution and entailment knowledge can be seen as complementary ways of achieving the same goal, namely enriching T with additional knowledge to allow the inference of H. Given that both of the technologies are still imperfect, we envisage the way forward as a joint strategy, where reference resolution and entailment rules mutually fill each other's gaps (cf. Example 3).

In sum, our study shows that textual entailment can profit substantially from better discourse handling. The next challenge is to translate the theoretical gain into practical benefit. Our analysis demonstrates that improvements are necessary both on the side of discourse reference resolution systems, which need to cover more types of references, as well as a better integration of discourse information in entailment systems, even for those relations which are within the scope of available resolvers.

Acknowledgements

This work was partially supported by the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886 and the Israel Science Foundation grant 1112/08.

References

Azad Abad, Luisa Bentivogli, Ido Dagan, Danilo Giampiccolo, Shachar Mirkin, Emanuele Pianta, and Asher Stern. 2010. A resource for investigating the impact of anaphora and coreference on inference. In *Proceedings of LREC*.

Rod Adams, Gabriel Nicolae, Cristina Nicolae, and Sanda Harabagiu. 2007. Textual entailment through extended lexical overlap and lexico-semantic matching. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

E. Agichtein, W. Askew, and Y. Liu. 2008. Combining lexical, syntactic, and semantic evidence for textual entailment classification. In *Proceedings of TAC*.

Nicholas Asher and Alex Lascarides. 1998. Bridging. *Journal of Semantics*, 15(1):83–113.

Alexandra Balahur, Elena Lloret, Óscar Ferrández, Andrés Montoyo, Manuel Palomar, and Rafael Muñoz. 2008. The DLSIUAES team's participation in the TAC 2008 tracks. In *Proceedings of TAC*.

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.

Roy Bar-Haim, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, and Eyal Shnarch amd Idan Szpektor. 2008. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of TAC*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009a. The fifth pascal recognizing textual entailment challenge. In *Proceedings of TAC*.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2009b. Considering discourse references in textual entailment annotation. In *Proceedings of the 5th International Conference on Generative Approaches to the Lexicon (GL2009)*.

Johan Bos. 2005. Recognising textual entailment with logical inference. In *Proceedings of EMNLP*.

Aljoscha Burchardt, Marco Pennacchiotti, Stefan Thater, and Manfred Pinkal. 2009. Assessing the impact of frame semantics on textual entailment. *Journal of Natural Language Engineering*, 15(4):527–550.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of ACL-IJCNLP*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Herbert H. Clark. 1975. Bridging. In R. C. Schank and B. L. Nash-Webber, editors, *Theoretical issues in natural language processing*, pages 169–174. Association of Computing Machinery.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Lorand Dali, Delia Rusu, Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. 2009. Question answering based on semantic graphs. In *Proceedings of the Workshop on Semantic Search (SemSearch 2009)*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2008. The fourth pascal recognizing textual entailment challenge. In *Proceedings of TAC*.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th conference on Computational Linguistics*.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2007. Satisfying information needs with multi-document summaries. *Information Processing & Management*, 43:1619–1642.

Stefan Harmeling. 2009. Inferring textual entailment with a probabilistically sound calculus. *Journal of Natural Language Engineering*, pages 459–477.

Marti A. Hearst. 1997. Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of HLT-NAACL*.

Jian Huang, Sarah M. Taylor, Jonathan L. Smith, Konstantinos A. Fotiadis, and C. Lee Giles. 2009. Profile based cross-document coreference using kernelized fuzzy relational clustering. In *Proceedings of ACL-IJCNLP*.

Fangtao Li, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering opinion questions with random walks on graphs. In *Proceedings of ACL-IJCNLP*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 4(7):343–360.

Dekang Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of ACL*.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

Katja Markert, Malvina Nissim, and Natalia N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of EACL Workshop on the Computational Treatment of Anaphora*.

Seung-Hoon Na and Hwee Tou Ng. 2009. A 2-poisson model for probabilistic coreference of named entities for improved text retrieval. In *Proceedings of SIGIR*.

Rodney D. Nielsen, Wayne Ward, and James H. Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(4):479–501.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of ACL*.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of HLT*.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2004. A public reference implementation of the rap anaphora resolution algorithm. In *Proceedings of LREC*.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Stephanie Strassel, Mark Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *Proceedings of LREC*.

Jose L. Vicedo and Antonio Ferrndez. 2006. Coreference in Q&A. In Tomek Strzalkowski and Sanda M. Harabagiu, editors, *Advances in Open Domain Question Answering*, pages 71–96. Springer.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Journal of Natural Language Engineering*, 15(4):551–582.

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbetts. 2004. Coreference resolution for information extraction. In *Proceedings of the ACL Workshop on Reference Resolution and its Applications*.

Global Learning of Focused Entailment Graphs

Jonathan Berant
Tel-Aviv University
Tel-Aviv, Israel
jonatha6@post.tau.ac.il

Ido Dagan
Bar-Ilan University
Ramat-Gan, Israel
dagan@cs.biu.ac.il

Jacob Goldberger
Bar-Ilan University
Ramat-Gan, Israel
goldbej@eng.biu.ac.il

Abstract

We propose a global algorithm for learning entailment relations between predicates. We define a graph structure over predicates that represents entailment relations as directed edges, and use a global transitivity constraint on the graph to learn the optimal set of edges, by formulating the optimization problem as an Integer Linear Program. We motivate this graph with an application that provides a hierarchical summary for a set of propositions that focus on a target concept, and show that our global algorithm improves performance by more than 10% over baseline algorithms.

1 Introduction

The *Textual Entailment (TE)* paradigm (Dagan et al., 2009) is a generic framework for applied semantic inference. The objective of TE is to recognize whether a target meaning can be inferred from a given text. For example, a Question Answering system has to recognize that *'alcohol affects blood pressure'* is inferred from *'alcohol reduces blood pressure'* to answer the question *'What affects blood pressure?'*

TE systems require extensive knowledge of entailment patterns, often captured as *entailment rules:* rules that specify a directional inference relation between two text fragments (when the rule is bidirectional this is known as paraphrasing). An important type of entailment rule refers to *propositional templates*, i.e., propositions comprising a predicate and arguments, possibly replaced by variables. The rule required for the previous example would be *'X reduce Y* $\rightarrow$ *X affect Y'*. Because facts and knowledge are mostly expressed by propositions, such entailment rules are central to the TE task. This has led to active research on broad-scale acquisition of entailment rules for predicates, e.g. (Lin and Pantel, 2001; Sekine, 2005; Szpektor and Dagan, 2008).

Previous work has focused on learning each entailment rule in isolation. However, it is clear that there are interactions between rules. A prominent example is that entailment is a transitive relation, and thus the rules '$X \rightarrow Y$' and '$Y \rightarrow Z$' imply the rule '$X \rightarrow Z$'. In this paper we take advantage of these global interactions to improve entailment rule learning.

First, we describe a structure termed an *entailment graph* that models entailment relations between propositional templates (Section 3). Next, we show that we can present propositions according to an entailment hierarchy derived from the graph, and suggest a novel hierarchical presentation scheme for corpus propositions referring to a target concept. As in this application each graph focuses on a single concept, we term those *focused entailment graphs* (Section 4).

In the core section of the paper, we present an algorithm that uses a global approach to learn the entailment relations of focused entailment graphs (Section 5). We define a global function and look for the graph that maximizes that function under a transitivity constraint. The optimization problem is formulated as an *Integer Linear Program (ILP)* and solved with an ILP solver. We show that this leads to an optimal solution with respect to the global function, and demonstrate that the algorithm outperforms methods that utilize only local information by more than 10%, as well as methods that employ a greedy optimization algorithm rather than an ILP solver (Section 6).

2 Background

Entailment learning Two information types have primarily been utilized to learn entailment rules between predicates: lexicographic resources and distributional similarity resources. Lexicographic

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1220–1229,
Uppsala, Sweden, 11-16 July 2010.

resources are manually-prepared knowledge bases containing information about semantic relations between lexical items. WordNet (Fellbaum, 1998), by far the most widely used resource, specifies relations such as *hyponymy*, *derivation*, and *entailment* that can be used for semantic inference (Budanitsky and Hirst, 2006). WordNet has also been exploited to automatically generate a training set for a hyponym classifier (Snow et al., 2005), and we make a similar use of WordNet in Section 5.1.

Lexicographic resources are accurate but tend to have low coverage. Therefore, distributional similarity is used to learn broad-scale resources. Distributional similarity algorithms predict a semantic relation between two predicates by comparing the arguments with which they occur. Quite a few methods have been suggested (Lin and Pantel, 2001; Bhagat et al., 2007; Yates and Etzioni, 2009), which differ in terms of the specifics of the ways in which predicates are represented, the features that are extracted, and the function used to compute feature vector similarity. Details on such methods are given in Section 5.1.

Global learning It is natural to describe entailment relations between predicates by a graph. Nodes represent predicates, and edges represent entailment between nodes. Nevertheless, using a graph for global learning of entailment between predicates has attracted little attention. Recently, Szpektor and Dagan (2009) presented the resource *Argument-mapped WordNet*, providing entailment relations for predicates in WordNet. Their resource was built on top of WordNet, and makes simple use of WordNet's global graph structure: new rules are suggested by transitively chaining graph edges, and verified against corpus statistics.

The most similar work to ours is Snow et al.'s algorithm for taxonomy induction (2006). Snow et al.'s algorithm learns the hyponymy relation, under the constraint that it is a *transitive relation*. Their algorithm incrementally adds hyponyms to an existing taxonomy (WordNet), using a greedy search algorithm that adds at each step the set of hyponyms that maximize the probability of the evidence while respecting the transitivity constraint.

In this paper we tackle a similar problem of learning a transitive relation, but we use linear programming. A *Linear Program (LP)* is an optimization problem, where a linear function is minimized (or maximized) under linear constraints. If the variables are integers, the problem is termed an *Integer Linear Program (ILP)*. Linear programming has attracted attention recently in several fields of NLP, such as semantic role labeling, summarization and parsing (Roth and tau Yih, 2005; Clarke and Lapata, 2008; Martins et al., 2009). In this paper we formulate the entailment graph learning problem as an Integer Linear Program, and find that this leads to an optimal solution with respect to the target function in our experiment.

3 Entailment Graph

This section presents an *entailment graph* structure, which resembles the graph in (Szpektor and Dagan, 2009).

The nodes of an entailment graph are *propositional templates*. A propositional template is a path in a dependency tree between two arguments of a common predicate[1] (Lin and Pantel, 2001; Szpektor and Dagan, 2008). Note that in a dependency parse, such a path passes through the predicate. We require that a variable appears in at least one of the argument positions, and that each sense of a polysemous predicate corresponds to a separate template (and a separate graph node): *X* $\xleftarrow{subj}$ *treat#1* $\xrightarrow{obj}$ *Y* and *X* $\xleftarrow{subj}$ *treat#1* $\xrightarrow{obj}$ *nausea* are propositional templates for the first sense of the predicate *treat*. An edge (u, v) represents the fact that template u entails template v. Note that the entailment relation transcends beyond hyponymy. For example, the template *X is diagnosed with asthma* entails the template *X suffers from asthma*, although one is not a hyponoym of the other. An example of an entailment graph is given in Figure 1, left.

Since entailment is a transitive relation, an entailment graph is *transitive*, i.e., if the edges (u, v) and (v, w) are in the graph, so is the edge (u, w). This is why we require that nodes be sense-specified, as otherwise transitivity does not hold: Possibly $a \rightarrow b$ for one sense of b, $b \rightarrow c$ for another sense of b, but $a \nrightarrow c$.

Because graph nodes represent propositions, which generally have a clear truth value, we can assume that transitivity is indeed maintained along paths of any length in an entailment graph, as entailment between each pair of nodes either occurs or doesn't occur with very high probability. We support this further in section 4.1, where we show

[1] We restrict our discussion to templates with two arguments, but generalization is straightforward.

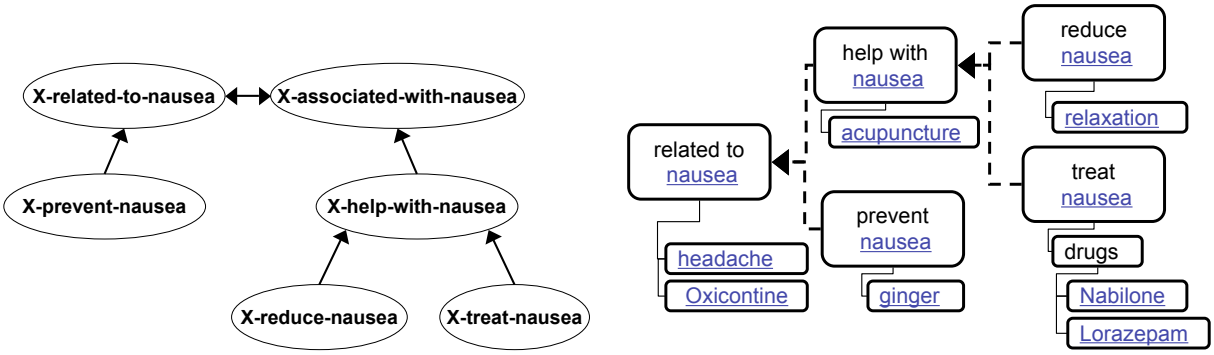


Figure 1: *Left:* An entailment graph. For clarity, edges that can be inferred by transitivity are omitted. *Right:* A hierarchical summary of propositions involving *nausea* as an argument, such as *headache is related to nausea*, *acupuncture helps with nausea*, and *Lorazepam treats nausea*.

that in our experimental setting the length of paths in the entailment graph is relatively small.

Transitivity implies that in each strong connectivity component[2] of the graph, all nodes are synonymous. Moreover, if we merge every strong connectivity component to a single node, the graph becomes a Directed Acyclic Graph (DAG), and the graph nodes can be sorted and presented hierarchically. Next, we show an application that leverages this property.

4 Motivating Application

In this section we propose an application that provides a hierarchical view of propositions extracted from a corpus, based on an entailment graph.

Organizing information in large collections has been found to be useful for effective information access (Kaki, 2005; Stoica et al., 2007). It allows for easier data exploration, and provides a compact view of the underlying content. A simple form of structural presentation is by a single hierarchy, e.g. (Hofmann, 1999). A more complex approach is hierarchical faceted metadata, where a number of concept hierarchies are created, corresponding to different facets or dimensions (Stoica et al., 2007).

Hierarchical faceted metadata categorizes *concepts* of a domain in several dimensions, but does not specify the *relations* between them. For example, in the health-care domain we might have facets for categories such as *diseases* and *symptoms*. Thus, when querying about *nausea*, one might find it is related to *vomitting* and *chicken pox*, but not that chicken pox is a *cause* of nausea, while nausea is often *accompanied* by vomitting.

We suggest that the prominent information in a text lies in the *propositions* it contains, which specify particular relations between the concepts. Propositions have been mostly presented through unstructured textual summaries or manually-constructed ontologies, which are expensive to build. We propose using the entailment graph structure, which describes entailment relations between predicates, to naturally present propositions hierarchically. That is, the entailment hierarchy can be used as an additional facet, which can improve navigation and provide a compact hierarchical summary of the propositions.

Figure 1 illustrates a scenario, on which we evaluate later our learning algorithm. Assume a user would like to retrieve information about a target concept such as *nausea*. We can extract the set of propositions where *nausea* is an argument automatically from a corprus, and learn an entailment graph over propositional templates derived from the extracted propositions, as illustrated in Figure 1, left. Then, we follow the steps in the process described in Section 3: merge synonymous nodes that are in the same strong connectivity component, and turn the resulting DAG into a predicate hierarchy, which we can then use to present the propositions (Figure 1, right). Note that in all propositional templates one argument is the target concept (nausea), and the other is a variable whose corpus instantiations can be presented according to another hierarchy (e.g. *Nabilone* and *Lorazepam* are types of *drugs*).

Moreover, new propositions are inferred from the graph by transitivity. For example, from the proposition *'relaxation reduces nausea'* we can in-

[2] A strong connectivity component is a subset of nodes in the graph where there is a path from any node to any other node.

fer the proposition *'relaxation helps with nausea'*.

4.1 Focused entailment graphs

The application presented above generates entailment graphs of a specific form: (1) Propositional templates have exactly one argument instantiated by the same entity (e.g. nausea). (2) The predicate sense is unspecified, but due to the rather small number of nodes and the instantiating argument, each predicate corresponds to a unique sense.

Generalizing this notion, we define a *focused entailment graph* to be an entailment graph where the number of nodes is relatively small (and consequently paths in the graph are short), and predicates have a single sense (so transitivity is maintained without sense specification). Section 5 presents an algorithm that given the set of nodes of a focused entailment graph learns its edges, i.e., the entailment relations between all pairs of nodes. The algorithm is evaluated in Section 6 using our proposed application. For brevity, from now on the term *entailment graph* will stand for *focused entailment graph*.

5 Learning Entailment Graph Edges

In this section we present an algorithm for learning the edges of an entailment graph given its set of nodes. The first step is preprocessing: We use a large corpus and WordNet to train an *entailment classifier* that estimates the likelihood that one propositional template entails another. Next, we can learn on the fly for any input graph: given the graph nodes, we employ a global optimization approach that determines the set of edges that maximizes the probability (or score) of the entire graph, given the edge probabilities (or scores) supplied by the entailment classifier and the graph constraints (transitivity and others).

5.1 Training an entailment classifier

We describe a procedure for learning an entailment classifier, given a corpus and a lexicographic resource (WordNet). First, we extract a large set of propositional templates from the corpus. Next, we represent each pair of propositional templates with a feature vector of various distributional similarity scores. Last, we use WordNet to automatically generate a training set and train a classifier.

Template extraction We parse the corpus with a dependency parser and extract all propositional templates from every parse tree, employing the procedure used by Lin and Pantel (2001). However, we only consider templates containing a predicate term and arguments[3]. The arguments are replaced with variables, resulting in propositional templates such as $X \xleftarrow{subj} affect \xrightarrow{obj} Y$.

Distributional similarity representation We aim to train a classifier that for an input template pair (t_1, t_2) determines whether t_1 entails t_2. A template pair is represented by a feature vector where each coordinate is a different distributional similarity score. There are a myriad of distributional similarity algorithms. We briefly describe those used in this paper, obtained through variations along the following dimensions:

Predicate representation Most algorithms measure the similarity between templates with two variables (*binary templates*) such as $X \xleftarrow{subj} affect \xrightarrow{obj} Y$ (Lin and Pantel, 2001; Bhagat et al., 2007; Yates and Etzioni, 2009). Szpketor and Dagan (2008) suggested learning over templates with one variable (*unary templates*) such as $X \xleftarrow{subj} affect$, and using them to estimate a score for binary templates.

Feature representation The features of a template are some representation of the terms that instantiated the argument variables in a corpus. Two representations are used in our experiment (see Section 6). Another variant occurs when using binary templates: a template may be represented by a pair of feature vectors, one for each variable (Lin and Pantel, 2001), or by a single vector, where features represent pairs of instantiations (Szpektor et al., 2004; Yates and Etzioni, 2009). The former variant reduces sparsity problems, while Yates and Etzioni showed the latter is more informative and performs favorably on their data.

Similarity function We consider two similarity functions: The *Lin* (2001) similarity measure, and the *Balanced Inclusion (BInc)* similarity measure (Szpektor and Dagan, 2008). The former is a symmetric measure and the latter is asymmetric. Therefore, information about the direction of entailment is provided by the BInc measure.

We then generate for any (t_1, t_2) features that are the 12 distributional similarity scores using all combinations of the dimensions. This is reminiscent of Connor and Roth (2007), who used the output of unsupervised classifiers as features for a supervised classifier in a verb disambiguation task.

[3] Via a simple heuristic, omitted due to space limitations

Training set generation Following the spirit of Snow et al. (2005), WordNet is used to automatically generate a training set of positive (entailing) and negative (non-entailing) template pairs. Let T be the set of propositional templates extracted from the corpus. For each $t_i \in T$ with two variables and a single predicate word w, we extract from WordNet the set H of direct hypernyms and synonyms of w. For every $h \in H$, we generate a new template t_j from t_i by replacing w with h. If $t_j \in T$, we consider (t_i, t_j) to be a positive example. Negative examples are generated analogously, by looking at direct co-hyponyms of w instead of hypernyms and synonyms. This follows the notion of "contrastive estimation" (Smith and Eisner, 2005), since we generate negative examples that are semantically similar to positive examples and thus focus the classifier's attention on identifying the boundary between the classes. Last, we filter training examples for which all features are zero, and sample an equal number of positive and negative examples (for which we compute similarity features), since classifiers tend to perform poorly on the minority class when trained on imbalanced data (Van Hulse et al., 2007; Nikulin, 2008).

5.2 Global learning of edges

Once the entailment classifier is trained we learn the graph edges given its nodes. This is equivalent to learning all entailment relations between all propositional template pairs for that graph.

To learn edges we consider global constraints, which allow only certain graph topologies. Since we seek a global solution under transitivity and other constraints, linear programming is a natural choice, enabling the use of state of the art optimization packages. We describe two formulations of integer linear programs that learn the edges: one maximizing a global *score* function, and another maximizing a global *probability* function.

Let I_{uv} be an indicator denoting the event that node u entails node v. Our goal is to learn the edges E over a set of nodes V. We start by formulating the constraints and then the target functions.

The first constraint is that the graph must respect transitivity. Our formulation is equivalent to the one suggested by Finkel and Manning (2008) in a coreference resolution task:

$$\forall_{u,v,w \in V} I_{uv} + I_{vw} - I_{uw} \leq 1$$

In addition, for a few pairs of nodes we have strong evidence that one does not entail the other and so we add the constraint $I_{uv} = 0$. Combined with the constraint of transitivity this implies that there must be no path from u to v. This is done in the following two scenarios: (1) When two nodes u and v are identical except for a pair of words w_u and w_v, and w_u is an antonym of w_v, or a hypernym of w_v at distance ≥ 2. (2) When two nodes u and v are transitive opposites, that is, if $u = X \xleftarrow{subj} w \xrightarrow{obj} Y$ and $v = X \xleftarrow{obj} w \xrightarrow{subj} Y$, for any word w[4].

Score-based target function We assume an entailment classifier estimating a positive score S_{uv} if it believes $I_{uv} = 1$ and a negative score otherwise (for example, an SVM classifier). We look for a graph G that maximizes the sum of scores over the edges:

$$\begin{aligned}\hat{G} &= \operatorname*{argmax}_G S(G) \\ &= \operatorname*{argmax}_G \left(\sum_{u \neq v} S_{uv} I_{uv} \right) - \lambda |E|\end{aligned}$$

where $\lambda|E|$ is a regularization term reflecting the fact that edges are sparse. Note that this constant needs to be optimized on a development set.

Probabilistic target function Let F_{uv} be the features for the pair of nodes (u, v) and $F = \cup_{u \neq v} F_{uv}$. We assume an entailment classifier estimating the probability of an edge given its features: $P_{uv} = P(I_{uv} = 1 | F_{uv})$. We look for the graph G that maximizes the posterior probability $P(G|F)$:

$$\hat{G} = \operatorname*{argmax}_G P(G|F)$$

Following Snow et al., we make two independence assumptions: First, we assume each set of features F_{uv} is independent of other sets of features given the graph G, i.e., $P(F|G) = \prod_{u \neq v} P(F_{uv}|G)$. Second, we assume the features for the pair (u, v) are generated by a distribution depending only on whether entailment holds for (u, v). Thus, $P(F_{uv}|G) = P(F_{uv}|I_{uv})$. Last, for simplicity we assume edges are independent and the prior probability of a graph is a product of the prior probabilities of the edge indicators:

[4] We note that in some rare cases transitive verbs are indeed reciprocal, as in *"X marry Y"*, but in the grand majority of cases reciprocal activities are not expressed using a transitive-verb structure.

$P(G) = \prod_{u \neq v} P(I_{uv})$. Note that although we assume edges are independent, dependency is still expressed using the transitivity constraint. We express $P(G|F)$ using the assumptions above and Bayes rule:

$$\begin{aligned} P(G|F) &\propto P(G)P(F|G) \\ &= \prod_{u \neq v} [P(I_{uv})P(F_{uv}|I_{uv})] \\ &= \prod_{u \neq v} P(I_{uv}) \frac{P(I_{uv}|F_{uv})P(F_{uv})}{P(I_{uv})} \\ &\propto \prod_{u \neq v} P(I_{uv}|F_{uv}) \\ &= \prod_{(u,v) \in E} P_{uv} \cdot \prod_{(u,v) \notin E} (1 - P_{uv}) \end{aligned}$$

Note that the prior $P(F_{uv})$ is constant with respect to the graph. Now we look for the graph that maximizes $\log P(G|F)$:

$$\begin{aligned} \hat{G} &= \operatorname*{argmax}_G \sum_{(u,v) \in E} \log P_{uv} + \sum_{(u,v) \notin E} \log(1 - P_{uv}) \\ &= \operatorname*{argmax}_G \sum_{u \neq v} [I_{uv} \cdot \log P_{uv} \\ &\qquad + (1 - I_{uv}) \cdot \log(1 - P_{uv})] \\ &= \operatorname*{argmax}_G \sum_{u \neq v} \log \frac{P_{uv}}{1 - P_{uv}} \cdot I_{uv} \end{aligned}$$

(in the last transition we omit the constant $\sum_{u \neq v} \log(1 - P_{uv})$). Importantly, while the score-based formulation contains a parameter λ that requires optimization, this probabilistic formulation is parameter free and does not utilize a development set at all.

Since the variables are binary, both formulations are integer linear programs with $O(|V|^2)$ variables and $O(|V|^3)$ transitivity constraints that can be solved using standard ILP packages.

Our work resembles Snow et al.'s in that both try to learn graph edges given a transitivity constraint. However, there are two key differences in the model and in the optimization algorithm. First, Snow et al.'s model attempts to determine the graph that maximizes the likelihood $P(F|G)$ and not the posterior $P(G|F)$. Therefore, their model contains an edge prior $P(I_{uv})$ that has to be estimated, whereas in our model it cancels out. Second, they incrementally add hyponyms to a large taxonomy (WordNet) and therefore utilize a greedy algorithm, while we simultaneously learn all edges of a rather small graph and employ integer linear programming, which is more sound theoretically, and as shown in Section 6, leads to an optimal solution. Nevertheless, Snow et al.'s model can also be formulated as a linear program with the following target function:

$$\operatorname*{argmax}_G \sum_{u \neq v} \log \frac{P_{uv} \cdot P(I_{uv} = 0)}{(1 - P_{uv}) \cdot P(I_{uv} = 1)} I_{uv}$$

Note that if the prior inverse odds $k = \frac{P(I_{uv}=0)}{P(I_{uv}=1)} = 1$, i.e., $P(I_{uv} = 1) = 0.5$, then this is equivalent to our probabilistic formulation. We implemented Snow et al's model and optimization algorithm and in Section 6.3 we compare our model and optimization algorithm to theirs.

6 Experimental Evaluation

This section presents our evaluation, which is geared for the application proposed in Section 4.

6.1 Experimental setting

A health-care corpus of 632MB was harvested from the web and parsed with the Minipar parser (Lin, 1998). The corpus contains 2,307,585 sentences and almost 50 million word tokens. We used the Unified Medical Language System (UMLS)[5] to annotate medical concepts in the corpus. The UMLS is a database that maps natural language phrases to over one million *concept identifiers* in the health-care domain (termed CUIs). We annotated all nouns and noun phrases that are in the UMLS with their possibly multiple CUIs. We extracted all propositional templates from the corpus, where both argument instantiations are medical concepts, i.e., annotated with a CUI (∼50,000 templates). When computing distributional similarity scores, a template is represented as a feature vector of the CUIs that instantiate its arguments.

To evaluate the performance of our algorithm, we constructed 23 gold standard entailment graphs. First, 23 medical concepts, representing typical topics of interest in the medical domain, were manually selected from a list of the most frequent concepts in the corpus. For each concept, nodes were defined by extracting all propositional

[5] http://www.nlm.nih.gov/research/umls

	Using a development set						Not using a development set					
	Edges			Propositions			Edges			Propositions		
	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
LP	46.0	50.1	**43.8**	67.3	69.6	**66.2**	48.7	41.9	**41.2**	67.9	62.0	62.3
Greedy	45.7	37.1	36.6	64.2	57.2	56.3	48.2	41.7	41.0	67.8	62.0	**62.4**
Local-LP	44.5	45.3	38.1	65.2	61.0	58.6	69.3	19.7	26.8	82.7	33.3	42.6
$Local_1$	53.5	34.9	37.5	73.5	50.6	56.1	92.9	11.1	19.7	95.4	18.6	30.6
$Local_2$	52.5	31.6	37.7	69.8	50.0	57.1	63.2	24.9	33.6	77.7	39.3	50.5
$Local_1^*$	53.5	38.0	39.8	73.5	54.6	59.1	92.6	11.3	20.0	95.3	18.9	31.1
$Local_2^*$	52.5	32.1	38.1	69.8	50.6	57.4	63.1	25.5	34.0	77.7	39.9	50.9
WordNet	-	-	-	-	-	-	10.8	44.1	13.2	39.9	72.4	47.3

Table 1: Results for all experiments

templates for which the target concept instantiated an argument at least $K(= 3)$ times (average number of graph nodes=22.04, std=3.66, max=26, min=13).

Ten medical students constructed the gold standard of graph edges. Each concept graph was annotated by two students. Following RTE-5 practice (Bentivogli et al., 2009), after initial annotation the two students met for a reconciliation phase. They worked to reach an agreement on differences and corrected their graphs. Inter-annotator agreement was calculated using the Kappa statistic (Siegel and Castellan, 1988) both before ($\kappa = 0.59$) and after ($\kappa = 0.9$) reconciliation. 882 edges were included in the 23 graphs out of a possible 10,364, providing a sufficiently large data set. The graphs were randomly split into a development set (11 graphs) and a test set (12 graphs)[6]. The entailment graph fragment in Figure 1 is from the gold standard.

The graphs learned by our algorithm were evaluated by two measures, one evaluating the graph directly, and the other motivated by our application: (1) F_1 of the learned edges compared to the gold standard edges (2) Our application provides a summary of propositions extracted from the corpus. Note that we infer new propositions by propagating inference transitively through the graph. Thus, we compute F_1 for the set of propositions inferred from the learned graph, compared to the set inferred based on the gold standard graph. For example, given the proposition from the corpus *'relaxation reduces nausea'* and the edge *'X reduce nausea → X help with nausea'*, we evaluate the set {*'relaxation reduces nausea', 'relaxation helps with nausea'*}. The final score for an algorithm is a macro-average over the 12 graphs of the test set.

[6]Test set concepts were: asthma, chemotherapy, diarrhea, FDA, headache, HPV, lungs, mouth, salmonella, seizure, smoking and X-ray.

6.2 Evaluated algorithms

Local algorithms We described 12 distributional similarity measures computed over our corpus (Section 5.1). For each measure we computed for each template t a list of templates most similar to t (or entailing t for directional measures). In addition, we obtained similarity lists learned by Lin and Pantel (2001), and replicated 3 similarity measures learned by Szpektor and Dagan (2008), over the RCV1 corpus[7]. For each distributional similarity measure (altogether 16 measures), we learned a graph by inserting any edge (u, v), when u is in the top K templates most similar to v. We also omitted edges for which there was strong evidence that they do not exist, as specified by the constraints in Section 5.2. Another local resource was WordNet where we inserted an edge (u, v) when v was a direct hypernym or synonym of u. For all algorithms, we added all edges inferred by transitivity.

Global algorithms We experimented with all 6 combinations of the following two dimensions: (1) Target functions: score-based, probabilistic and Snow et al.'s (2) Optimization algorithms: Snow et al.'s greedy algorithm and a standard ILP solver. A training set of 20,144 examples was automatically generated, each example represented by 16 features using the distributional similarity measures mentioned above. SVMperf (Joachims, 2005) was used to train an SVM classifier yielding S_{uv}, and the SMO classifier from WEKA (Hall et al., 2009) estimated P_{uv}. We used the *lpsolve*[8] package to solve the linear programs. In all results, the relaxation $\forall_{u,v} 0 \leq I_{uv} \leq 1$ was used, which guarantees an optimal output solution. In

[7]http://trec.nist.gov/data/reuters/reuters.html. The similarity lists were computed using: (1) Unary templates and the Lin function (2) Unary templates and the BInc function (3) Binary templates and the Lin function

[8]http://lpsolve.sourceforge.net/5.5/

	Global=T/Local=F	Global=F/Local=T
GS= T	50	143
GS= F	140	1087

Table 2: Comparing disagreements between the best local and global algorithms against the gold standard

all experiments the output solution was integer, and therefore it is optimal. Constructing graph nodes and learning its edges given an input concept took 2-3 seconds on a standard desktop.

6.3 Results and analysis

Table 1 summarizes the results of the algorithms. The left half depicts methods where the development set was needed to tune parameters, and the right half depicts methods that do not require a (manually created) development set at all. Hence, our score-based LP (*tuned-LP)*, where the parameter λ is tuned, is on the left, and the probabilistic LP (*untuned-LP*) is on the right. The row *Greedy* is achieved by using the greedy algorithm instead of *lpsolve*. The row *Local-LP* is achieved by omitting global transitivity constraints, making the algorithm completely local. We omit Snow et al.'s formulation, since the optimal prior inverse odds k was almost exactly 1, which conflates with untuned-LP.

The rows $Local_1$ and $Local_2$ present the best distributional similarity resources. Local_1 is achieved using binary templates, the *Lin* function, and a single vector with feature pairs. Local_2 is identical but employs the *BInc* function. $Local_1^*$ and $Local_2^*$ also exploit the local constraints mentioned above. Results on the left were achieved by optimizing the top-K parameter on the development set, and on the right by optimizing on the training set automatically generated from WordNet.

The global methods clearly outperform local methods: Tuned-LP outperforms significantly all local methods that require a development set both on the edges F_1 measure ($p<.05$) and on the propositions F_1 measure ($p<.01$)[9]. The untuned-LP algorithm also significantly outperforms all local methods that do not require a development set on the edges F_1 measure ($p<.05$) and on the propositions F_1 measure ($p<.01$). Omitting the global transitivity constraints decreases performance, as shown by *Local-LP*. Last, local methods are sensitive to parameter tuning and in the absence of a development set their performance dramatically deteriorates.

[9]We tested significance using the two-sided Wilcoxon rank test (Wilcoxon, 1945)

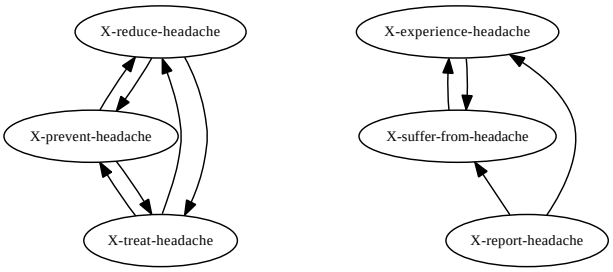


Figure 2: Subgraph of tuned-LP output for "headache"

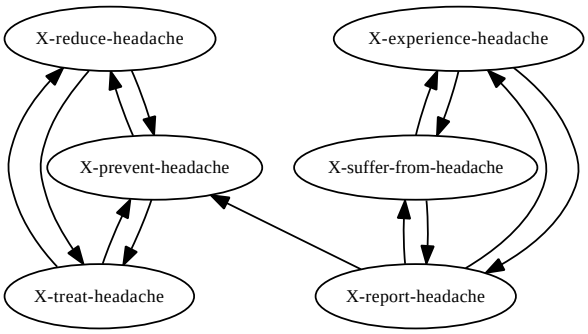


Figure 3: Subgraph of Local_1^* output for"headache"

To further establish the merits of global algorithms, we compare (Table 2) tuned-LP, the best global algorithm, with Local_1^*, the best local algorithm. The table considers all edges where the two algorithms disagree, and counts how many are in the gold standard and how many are not. Clearly, tuned-LP is superior at avoiding wrong edges (false positives). This is because tuned-LP refrains from adding edges that subsequently induce many undesirable edges through transitivity. Figures 2 and 3 illustrate this by comparing tuned-LP and Local_1^* on a subgraph of the *Headache* concept, before adding missing edges to satisfy transitivity to Local_1^* . Note that Local_1^* inserts a single wrong edge *X-report-headache* $\rightarrow$ *X-prevent-headache*, which leads to adding 8 more wrong edges. This is the type of global consideration that is addressed in an ILP formulation, but is ignored in a local approach and often overlooked when employing a greedy algorithm. Figure 2 also illustrates the utility of a local entailment graph for information presentation. Presenting information according to this subgraph distinguishes between propositions dealing with headache treatments and

propositions dealing with headache risk groups.

Comparing our use of an ILP algorithm to the greedy one reveals that tuned-LP significantly outperforms its greedy counterpart on both measures ($p<.01$). However, untuned-LP is practically equivalent to its greedy counterpart. This indicates that in this experiment the greedy algorithm provides a good approximation for the optimal solution achieved by our LP formulation.

Last, when comparing WordNet to local distributional similarity methods, we observe low recall and high precision, as expected. However, global methods achieve much higher recall than WordNet while maintaining comparable precision.

The results clearly demonstrate that a global approach improves performance on the entailment graph learning task, and the overall advantage of employing an ILP solver rather than a greedy algorithm.

7 Conclusion

This paper presented a global optimization algorithm for learning entailment relations between predicates represented as propositional templates. We modeled the problem as a graph learning problem, and searched for the best graph under a global transitivity constraint. We used Integer Linear Programming to solve the optimization problem, which is theoretically sound, and demonstrated empirically that this method outperforms local algorithms as well as a greedy optimization algorithm on the graph learning task.

Currently, we are investigating a generalization of our probabilistic formulation that includes a prior on the edges, and the relation of this prior to the regularization term introduced in our score-based formulation. In future work, we would like to learn general entailment graphs over a large number of nodes. This will introduce a challenge to our current optimization algorithm due to complexity issues, and will require careful handling of predicate ambiguity. Additionally, we will investigate novel features for the entailment classifier. This paper used distributional similarity, but other sources of information are likely to improve performance further.

Acknowledgments

We would like to thank Roy Bar-Haim, David Carmel and the anonymous reviewers for their useful comments. We also thank Dafna Berant and the nine students who prepared the gold standard data set. This work was developed under the collaboration of FBK-irst/University of Haifa and was partially supported by the Israel Science Foundation grant 1112/08. The first author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship, and has carried out this research in partial fulllment of the requirements for the Ph.D. degree.

References

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernarde Magnini. 2009. The fifth Pascal recognizing textual entailment challenge. In *Proceedings of TAC-09*.

Rahul Bhagat, Patrick Pantel, and Eduard Hovy. 2007. LEDIR: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of EMNLP-CoNLL*.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:273–381.

Michael Connor and Dan Roth. 2007. Context sensitive paraphrasing with a single unsupervised classifier. In *Proceedings of ECML*.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):1–17.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-08: HLT, Short Papers*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Thomas Hofmann. 1999. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *Proceedings of IJCAI*.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML*.

Mika Kaki. 2005. Findex: Search results categories help users when document ranking fails. In *Proceedings of CHI*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC*.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of ACL*.

Vladimir Nikulin. 2008. Classification of imbalanced data with random sets and mean-variance filtering. *IJDWM*, 4(2):63–78.

Dan Roth and Wen tau Yih. 2005. Integer linear programming inference for conditional random fields. In *Proceedings of ICML*, pages 737–744.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of IWP*.

Sideny Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New-York.

Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ACL*.

Emilia Stoica, Marti Hearst, and Megan Richardson. 2007. Automating creation of hierarchical faceted metadata structures. In *Proceedings of NAACL-HLT*.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of COLING*.

Idan Szpektor and Ido Dagan. 2009. Augmenting wordnet-based inference with argument mapping. In *Proceedings of TextInfer-2009*.

Idan Szpektor, Hristo Tanev, Ido Dagan, and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*.

Jason Van Hulse, Taghi Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of ICML*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Modeling Semantic Relevance for Question-Answer Pairs in Web Social Communities

Baoxun Wang, Xiaolong Wang, Chengjie Sun, Bingquan Liu, Lin Sun
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
{bxwang, wangxl, cjsun, liubq, lsun}@insun.hit.edu.cn

Abstract

Quantifying the semantic relevance between questions and their candidate answers is essential to answer detection in social media corpora. In this paper, a deep belief network is proposed to model the semantic relevance for question-answer pairs. Observing the textual similarity between the community-driven question-answering (cQA) dataset and the forum dataset, we present a novel learning strategy to promote the performance of our method on the social community datasets without hand-annotating work. The experimental results show that our method outperforms the traditional approaches on both the cQA and the forum corpora.

1 Introduction

In natural language processing (NLP) and information retrieval (IR) fields, question answering (QA) problem has attracted much attention over the past few years. Nevertheless, most of the QA researches mainly focus on locating the exact answer to a given factoid question in the related documents. The most well known international evaluation on the factoid QA task is the Text REtrieval Conference (TREC)[1], and the annotated questions and answers released by TREC have become important resources for the researchers. However, when facing a non-factoid question such as *why*, *how*, or *what about*, however, almost no automatic QA systems work very well.

The user-generated question-answer pairs are definitely of great importance to solve the non-factoid questions. Obviously, these natural QA pairs are usually created during people's communication via Internet social media, among which we are interested in the community-driven question-answering (cQA) sites and online forums. The cQA sites (or systems) provide platforms where users can either ask questions or deliver answers, and best answers are selected manually (e.g., Baidu Zhidao[2] and Yahoo! Answers[3]). Comparing with cQA sites, online forums have more virtual society characteristics, where people hold discussions in certain domains, such as techniques, travel, sports, etc. Online forums contain a huge number of QA pairs, and much noise information is involved.

To make use of the QA pairs in cQA sites and online forums, one has to face the challenging problem of distinguishing the questions and their answers from the noise. According to our investigation, the data in the community based sites, especially for the forums, have two obvious characteristics: (a) a post usually includes a very short content, and when a person is initializing or replying a post, an informal tone tends to be used; (b) most of the posts are useless, which makes the community become a noisy environment for question-answer detection.

In this paper, a novel approach for modeling the semantic relevance for QA pairs in the social media sites is proposed. We concentrate on the following two problems:

1. How to model the semantic relationship between two short texts using simple textual features? As mentioned above, the user generated questions and their answers via social media are always short texts. The limitation of length leads to the sparsity of the word features. In addition, the word frequency is usually either 0 or 1, that is, the frequency offers little information except the occurrence of a word. Because of this situation, the traditional relevance computing methods based on word co-occurrence, such as Cosine similarity and KL-divergence, are not effective for question-

[1] http://trec.nist.gov

[2] http://zhidao.baidu.com

[3] http://answers.yahoo.com

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1230–1238,
Uppsala, Sweden, 11-16 July 2010.

answer semantic modeling. Most researchers try to introduce structural features or users' behavior to improve the models performance, by contrast, the effect of textual features is not obvious.

2. How to train a model so that it has good performance on both cQA and forum datasets? So far, people have been doing QA researches on the cQA and the forum datasets separately (Ding et al., 2008; Surdeanu et al., 2008), and no one has noticed the relationship between the two kinds of data. Since both the cQA systems and the online forums are open platforms for people to communicate, the QA pairs in the cQA systems have similarity with those in the forums. In this case, it is highly valuable and desirable to propose a training strategy to improve the model's performance on both of the two kinds of datasets. In addition, it is possible to avoid the expensive and arduous hand-annotating work by introducing the method.

To solve the first problem, we present a deep belief network (DBN) to model the semantic relevance between questions and their answers. The network establishes the semantic relationship for QA pairs by minimizing the answer-to-question reconstructing error. Using only word features, our model outperforms the traditional methods on question-answer relevance calculating.

For the second problem, we make our model to learn the semantic knowledge from the solved question threads in the cQA system. Instead of mining the structure based features from cQA pages and forum threads individually, we consider the textual similarity between the two kinds of data. The semantic information learned from cQA corpus is helpful to detect answers in forums, which makes our model show good performance on social media corpora. Thanks to the labels for the best answers existing in the threads, no manual work is needed in our strategy.

The rest of this paper is organized as follows: Section 2 surveys the related work. Section 3 introduces the deep belief network for answer detection. In Section 4, the homogenous data based learning strategy is described. Experimental result is given in Section 5. Finally, conclusions and future directions are drawn in Section 6.

2 Related Work

The value of the naturally generated question-answer pairs has not been recognized until recent years. Early studies mainly focus on extracting QA pairs from frequently asked questions (FAQ) pages (Jijkoun and de Rijke, 2005; Riezler et al., 2007) or service call-center dialogues (Berger et al., 2000).

Judging whether a candidate answer is semantically related to the question in the cQA page automatically is a challenging task. A framework for predicting the quality of answers has been presented in (Jeon et al., 2006). Bernhard and Gurevych (2009) have developed a translation based method to find answers. Surdeanu et al. (2008) propose an approach to rank the answers retrieved by Yahoo! Answers. Our work is partly similar to Surdeanu et al. (2008), for we also aim to rank the candidate answers reasonably, but our ranking algorithm needs only word information, instead of the combination of different kinds of features.

Because people have considerable freedom to post on forums, there are a great number of irrelevant posts for answering questions, which makes it more difficult to detect answers in the forums. In this field, exploratory studies have been done by Feng et al. (2006) and Huang et al. (2007), who extract input-reply pairs for the discussion-bot. Ding et al.(2008) and Cong et al.(2008) have also presented outstanding research works on forum QA extraction. Ding et al. (2008) detect question contexts and answers using the conditional random fields, and a ranking algorithm based on the authority of forum users is proposed by Cong et al. (2008). Treating answer detection as a binary classification problem is an intuitive idea, thus there are some studies trying to solve it from this view (Hong and Davison, 2009; Wang et al., 2009). Especially Hong and Davison (2009) have achieved a rather high precision on the corpora with less noise, which also shows the importance of "social" features.

In order to select the answers for a given question, one has to face the problem of lexical gap. One of the problems with lexical gap embedding is to find similar questions in QA achieves (Jeon et al., 2005). Recently, the statistical machine translation (SMT) strategy has become popular. Lee et al. (2008) use translate models to bridge the lexical gap between queries and questions in QA collections. The SMT based methods are effective on modeling the semantic relationship between questions and answers and expending users' queries in answer retrieval (Riezler et al., 2007; Berger et al.,

2000; Bernhard and Gurevych, 2009). In (Surdeanu et al., 2008), the translation model is used to provide features for answer ranking.

The structural features (e.g., authorship, acknowledgement, post position, etc), also called non-textual features, play an important role in answer extraction. Such features are used in (Ding et al., 2008; Cong et al., 2008), and have significantly improved the performance. The studies of Jeon et al. (2006) and Hong et al. (2009) show that the structural features have even more contribution than the textual features. In this case, the mining of textual features tends to be ignored.

There are also some other research topics in this field. Cong et al. (2008) and Wang et al. (2009) both propose the strategies to detect questions in the social media corpus, which is proved to be a non-trivial task. The deep research on question detection has been taken by Duan et al. (2008). A graph based algorithm is presented to answer opinion questions (Li et al., 2009). In email summarization field, the QA pairs are also extracted from email contents as the main elements of email summarization (Shrestha and McKeown, 2004).

3 The Deep Belief Network for QA pairs

Due to the feature sparsity and the low word frequency of the social media corpus, it is difficult to model the semantic relevance between questions and answers using only co-occurrence features. It is clear that the semantic link exists between the question and its answers, even though they have totally different lexical representations. Thus a specially designed model may learn semantic knowledge by reconstructing a great number of questions using the information in the corresponding answers. In this section, we propose a deep belief network for modeling the semantic relationship between questions and their answers. Our model is able to map the QA data into a low-dimensional semantic-feature space, where a question is close to its answers.

3.1 The Restricted Boltzmann Machine

An ensemble of binary vectors can be modeled using a two-layer network called a "restricted Boltzmann machine" (RBM) (Hinton, 2002). The dimension reducing approach based on RBM initially shows good performance on image processing (Hinton and Salakhutdinov, 2006). Salakhutdinov and Hinton (2009) propose a deep graphical model composed of RBMs into the information retrieval field, which shows that this model is able to obtain semantic information hidden in the word-count vectors.

As shown in Figure 1, the RBM is a two-layer network. The bottom layer represents a visible vector **v** and the top layer represents a latent feature **h**. The matrix W contains the symmetric interaction terms between the visible units and the hidden units. Given an input vector **v**, the trained

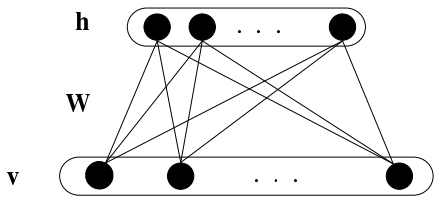


Figure 1: Restricted Boltzmann machine

RBM model provides a hidden feature **h**, which can be used to reconstruct **v** with a minimum error. The training algorithm for this paper will be described in the next subsection. The ability of the RBM suggests us to build a deep belief network based on RBM so that the semantic relevance between questions and answers can be modeled.

3.2 Pretraining a Deep Belief Network

In the social media corpora, the answers are always descriptive, containing one or several sentences. Noticing that an answer has strong semantic association with the question and involves more information than the question, we propose to train a deep belief network by reconstructing the question using its answers. The training object is to minimize the error of reconstruction, and after the pretraining process, a point that lies in a good region of parameter space can be achieved.

Firstly, the illustration of the DBN model is given in Figure 2. This model is composed of three layers, and here each layer stands for the RBM or its variant. The bottom layer is a variant form of RBM's designed for the QA pairs. This layer we design is a little different from the classical RBM's, so that the bottom layer can generate the hidden features according to the visible answer vector and reconstruct the question vector using the hidden features. The pre-training procedure of this architecture is practically convergent. In the bottom layer, the binary feature vectors based on the statistics of the word occurrence in the answers are used to compute the "hidden features" in the

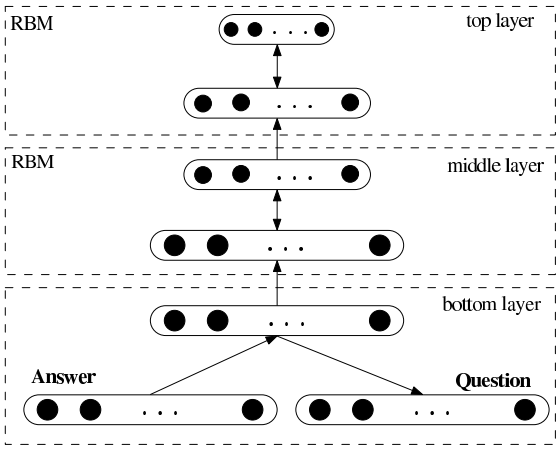


Figure 2: The Deep Belief Network for QA Pairs

hidden units. The model can reconstruct the questions using the hidden features. The processes can be modeled as follows:

$$p(h_j = 1|\mathbf{a}) = \sigma(b_j + \sum_i w_{ij} a_i) \quad (1)$$

$$p(q_i = 1|\mathbf{h}) = \sigma(b_i + \sum_j w_{ij} h_j) \quad (2)$$

where $\sigma(x) = 1/(1 + e^{-x})$, $\mathbf{a}$ denotes the visible feature vector of the answer, q_i is the *ith* element of the question vector, and $\mathbf{h}$ stands for the hidden feature vector for reconstructing the questions. w_{ij} is a symmetric interaction term between word i and hidden feature j, b_i stands for the bias of the model for word i, and b_j denotes the bias of hidden feature j.

Given the training set of answer vectors, the bottom layer generates the corresponding hidden features using Equation 1. Equation 2 is used to reconstruct the Bernoulli rates for each word in the question vectors after stochastically activating the hidden features. Then Equation 1 is taken again to make the hidden features active. We use 1-step Contrastive Divergence (Hinton, 2002) to update the parameters by performing gradient ascent:

$$\Delta w_{ij} = \epsilon(< q_i h_j >_{qData} - < q_i h_j >_{qRecon}) \quad (3)$$

where $< q_i h_j >_{qData}$ denotes the expectation of the frequency with which the word i in a question and the feature j are on together when the hidden features are driven by the question data. $< q_i h_j >_{qRecon}$ defines the corresponding expectation when the hidden features are driven by the reconstructed question data. ϵ is the learning rate.

The classical RBM structure is taken to build the middle layer and the top layer of the network. The training method for the higher two layer is similar to that of the bottom one, and we only have to make each RBM to reconstruct the input data using its hidden features. The parameter updates still obeying the rule defined by gradient ascent, which is quite similar to Equation 3. After training one layer, the $\mathbf{h}$ vectors are then sent to the higher-level layer as its "training data".

3.3 Fine-tuning the Weights

Notice that a greedy strategy is taken to train each layer individually during the pre-training procedure, it is necessary to fine-tune the weights of the entire network for optimal reconstruction. To fine-tune the weights, the network is unrolled, taking the answers as the input data to generate the corresponding questions at the output units. Using the cross-entropy error function, we can then tune the network by performing backpropagation through it. The experiment results in section 5.2 will show fine-tuning makes the network performs better for answer detection.

3.4 Best answer detection

After pre-training and fine-tuning, a deep belief network for QA pairs is established. To detect the best answer to a given question, we just have to send the vectors of the question and its candidate answers into the input units of the network and perform a level-by-level calculation to obtain the corresponding feature vectors. Then we calculate the distance between the mapped question vector and each candidate answer vector. We consider the candidate answer with the smallest distance as the best one.

4 Learning with Homogenous Data

In this section, we propose our strategy to make our DBN model to detect answers in both cQA and forum datasets, while the existing studies focus on one single dataset.

4.1 Homogenous QA Corpora from Different Sources

Our motivation of finding the homogenous question-answer corpora from different kind of social media is to guarantee the model's performance and avoid hand-annotating work.

In this paper, we get the "solved question" pages in the computer technology domain from Baidu Zhidao as the cQA corpus, and the threads of

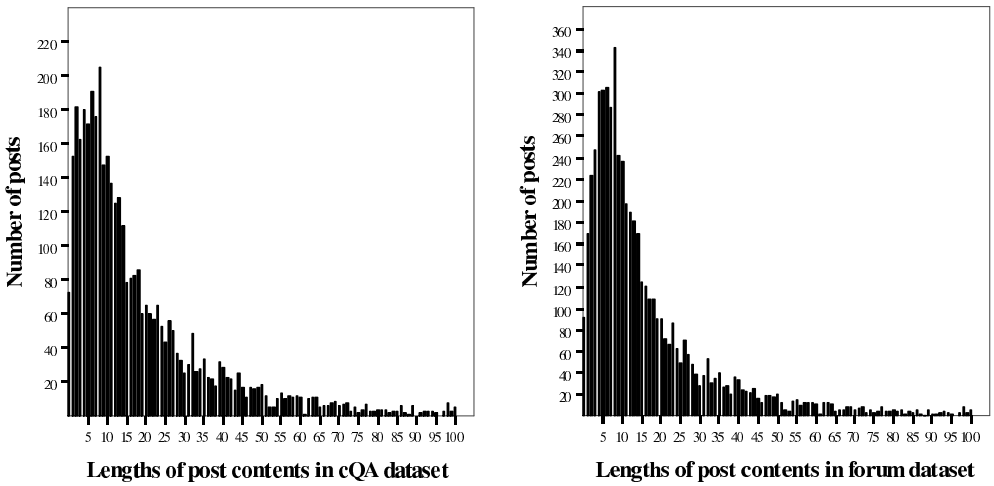


Figure 3: Comparison of the post content lengths in the cQA and the forum datasets

ComputerFansClub Forum[4] as the online forum corpus. The domains of the corpora are the same. To further explain that the two corpora are homogenous, we will give the detail comparison on text style and word distribution.

As shown in Figure 3, we have compared the post content lengths of the cQA and the forum in our corpora. For the comparison, 5,000 posts from the cQA corpus and 5,000 posts from the forum corpus are randomly selected. The left panel shows the statistical result on the Baidu Zhidao data, and the right panel shows the one on the forum data. The number i on the horizontal axis denotes the post contents whose lengths range from $10(i-1)+1$ to $10i$ bytes, and the vertical axis represents the counts of the post contents. From Figure 3 we observe that the contents of most posts in both the cQA corpus and the forum corpus are short, with the lengths not exceeding 400 bytes.

The content length reflects the text style of the posts in cQA systems and online forums. From Figure 3 it can be also seen that the distributions of the content lengths in the two figures are very similar. It shows that the contents in the two corpora are both mainly short texts.

Figure 4 shows the percentage of the concurrent words in the top-ranked content words with high frequency. In detail, we firstly rank the words by frequency in the two corpora. The words are chosen based on a professional dictionary to guarantee that they are meaningful in the computer knowledge field. The number k on the horizontal axis in Figure 4 represents the top k content words in the corpora, and the vertical axis stands for the percentage of the words shared by the two corpora in the top k words.

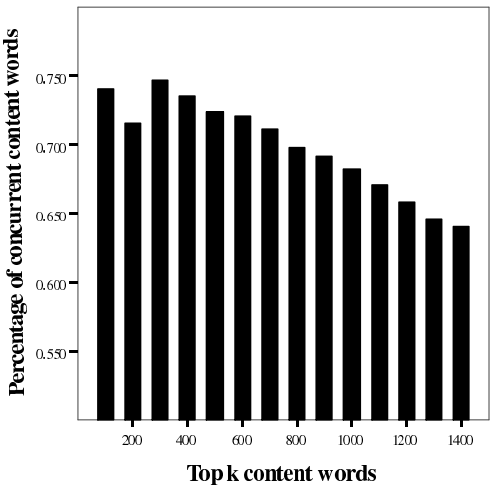


Figure 4: Distribution of concurrent content words

Figure 4 shows that a large number of meaningful words appear in both of the two corpora with high frequencies. The percentage of the concurrent words maintains above 64% in the top 1,400 words. It indicates that the word distributions of the two corpora are quite similar, although they come from different social media sites.

Because the cQA corpus and the forum corpus used in this study have homogenous characteristics for answer detecting task, a simple strategy may be used to avoid the hand-annotating work. Apparently, in every "solved question" page of Baidu Zhidao, the best answer is selected by the user who asks this question. We can easily extract the QA pairs from the cQA corpus as the training

[4]http://bbs.cfanclub.net/

set. Because the two corpora are similar, we can apply the deep belief network trained by the cQA corpus to detect answers on both the cQA data and the forum data.

4.2 Features

The task of detecting answers in social media corpora suffers from the problem of feature sparsity seriously. High-dimensional feature vectors with only several non-zero dimensions bring large time consumption to our model. Thus it is necessary to reduce the dimension of the feature vectors.

In this paper, we adopt two kinds of word features. Firstly, we consider the 1,300 most frequent words in the training set as Salakhutdinov and Hinton (2009) did. According to our statistics, the frequencies of the rest words are all less then 10, which are not statistically significant and may introduce much noise.

We take the occurrence of some function words as another kind of features. The function words are quite meaningful for judging whether a short text is an answer or not, especially for the non-factoid questions. For example, in the answers to the causation questions, the words such as *because* and *so* are more likely to appear; and the words such as *firstly*, *then*, and *should* may suggest the answers to the manner questions. We give an example for function word selection in Figure 5.

> ***Q***: 我的电脑开机一直黑屏，无法进 bios，应该怎么办？(The screen of my computer turns black after booting, and I can not enter bios, what should I do?)
> ***A***: ***可以先***拔掉硬盘，试试开机，***如果***能开机，***然后***在 bios 里把光驱设置成第一启动，重装系统吧。 (You ***may firstly*** unplug the hard disc and try to boot, ***if*** it works, ***then*** you ***should*** set your CD-Rom as the boot driver in bios and reinstall the system.)

Figure 5: An example for function word selection

For this reason, we collect 200 most frequent function words in the answers of the training set. Then for every short text, either a question or an answer, a 1,500-dimensional vector can be generated. Specifically, all the features we have adopted are binary, for they only have to denote whether the corresponding word appears in the text or not.

5 Experiments

To evaluate our question-answer semantic relevance computing method, we compare our approach with the popular methods on the answer detecting task.

5.1 Experiment Setup

Architecture of the Network: To build the deep belief network, we use a 1500-1500-1000-600 architecture, which means the three layers of the network have individually 1,500×1,500, 1,500×1,000 and 1,000×600 units. Using the network, a 1,500-dimensional binary vector is finally mapped to a 600-dimensional real-value vector.

During the pretraining stage, the bottom layer is greedily pretrained for 200 passes through the entire training set, and each of the rest two layers is greedily pretrained for 50 passes. For fine-tuning we apply the method of conjugate gradients[5], with three line searches performed in each pass. This algorithm is performed for 50 passes to fine-tune the network.

Dataset: we have crawled 20,000 pages of "solved question" from the *computer and network* category of Baidu Zhidao as the cQA corpus. Correspondingly we obtain 90,000 threads from ComputerFansClub, which is an online forum on computer knowledge. We take the forum threads as our forum corpus.

From the cQA corpus, we extract 12,600 human generated QA pairs as the training set without any manual work to label the best answers. We get the contents from another 2,000 cQA pages to form a testing set, each content of which includes one question and 4.5 candidate answers on average, with one best answer among them. To get another testing dataset, we randomly select 2,000 threads from the forum corpus. For this training set, human work are necessary to label the best answers in the posts of the threads. There are 7 posts included in each thread on average, among which one question and at least one answer exist.

Baseline: To show the performance of our method, three main popular relevance computing methods for ranking candidate answers are considered as our baselines. We will briefly introduce them:

Cosine Similarity. Given a question $\mathbf{q}$ and its candidate answer $\mathbf{a}$, their cosine similarity can be computed as follows:

$$\cos(\mathbf{q}, \mathbf{a}) = \frac{\sum_{k=1}^{n} w_{q_k} \times w_{a_k}}{\sqrt{\sum_{k=1}^{n} w_{q_k}^2} \times \sqrt{\sum_{k=1}^{n} w_{a_k}^2}} \quad (4)$$

where w_{q_k} and w_{a_k} stand for the weight of the kth word in the question and the answer respectively.

[5]Code is available at http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/

The weights can be get by computing the product of term frequency (*tf*) and inverse document frequency (*idf*)

HowNet based Similarity. HowNet[6] is an electronic world knowledge system, which serves as a powerful tool for meaning computation in human language technology. Normally the similarity between two passages can be calculated by two steps: (1) matching the most semantic-similar words in each passages greedily using the API's provided by HowNet; (2) computing the weighted average similarities of the word pairs. This strategy is taken as a baseline method for computing the relevance between questions and answers.

KL-divergence Language Model. Given a question **q** and its candidate answer **a**, we can construct unigram language model M_q and unigram language model M_a. Then we compute KL-divergence between M_q and M_a as below:

$$KL(M_a||M_q) = \sum_w p(w|M_a) \log(p(w|M_a)/p(w|M_q)) \quad (5)$$

5.2 Results and Analysis

We evaluate the performance of our approach for answer detection using two metrics: Precision@1 (P@1) and Mean Reciprocal Rank (MRR). Applying the two metrics, we perform the baseline methods and our DBN based methods on the two testing set above.

Table 1 lists the results achieved on the forum data using the baseline methods and ours. The additional "Nearest Answer" stands for the method without any ranking strategies, which returns the nearest candidate answer from the question by position. To illustrate the effect of the fine-tuning for our model, we list the results of our method without fine-tuning and the results with fine-tuning.

As shown in Table 1, our deep belief network based methods outperform the baseline methods as expected. The main reason for the improvements is that the DBN based approach is able to learn semantic relationship between the words in QA pairs from the training set. Although the training set we offer to the network comes from a different source (the cQA corpus), it still provide enough knowledge to the network to perform better than the baseline methods. This phenomena indicates that the homogenous corpora for training is effective and meaningful.

Method	P@1 (%)	MRR (%)
Nearest Answer	21.25	38.72
Cosine Similarity	23.15	43.50
HowNet	22.55	41.63
KL divergence	25.30	51.40
DBN (without FT)	**41.45**	**59.64**
DBN (with FT)	**45.00**	**62.03**

Table 1: Results on Forum Dataset

We have also investigated the reasons for the unsatisfying performance of the baseline approaches. Basically, the low precision is ascribable to the forum corpus we have obtained. As mentioned in Section 1, the contents of the forum posts are short, which leads to the sparsity of the features. Besides, when users post messages in the online forums, they are accustomed to be casual and use some synonymous words interchangeably in the posts, which is believed to be a significant situation in Chinese forums especially. Because the features for QA pairs are quite sparse and the content words in the questions are usually morphologically different from the ones with the same meaning in the answers, the Cosine Similarity method become less powerful. For HowNet based approaches, there are a large number of words not included by HowNet, thus it fails to compute the similarity between questions and answers. KL-divergence suffers from the same problems with the Cosine Similarity method. Compared with the Cosine Similarity method, this approach has achieved the improvement of 9.3% in P@1, but it performs much better than the other baseline methods in MRR.

The baseline results indicate that the online forum is a complex environment with large amount of noise for answer detection. Traditional IR methods using pure textual features can hardly achieve good results. The similar baseline results for forum answer ranking are also achieved by Hong and Davison (2009), which takes some non-textual features to improve the algorithm's performance. We also notice that, however, the baseline methods have obtained better results on forum corpus (Cong et al., 2008). One possible reason is that the baseline approaches are suitable for their data, since we observe that the "nearest answer" strategy has obtained a 73.5% precision in their work.

Our model has achieved the precision of

[6]Detail information can be found in: http://www.keenage.com/

45.00% in P@1 and 62.03% in MRR for answer detecting on forum data after fine-tuning, while some related works have reported the results with the precision over 90% (Cong et al., 2008; Hong and Davison, 2009). There are mainly two reasons for this phenomena: Firstly, both of the previous works have adopt non-textual features based on the forum structure, such as *authorship*, *position* and *quotes*, etc. The non-textual (or social based) features have played a significant role in improving the algorithms' performance. Secondly, the quality of corpora influences the results of the ranking strategies significantly, and even the same algorithm may perform differently when the dataset is changed (Hong and Davison, 2009). For the experiments of this paper, large amount of noise is involved in the forum corpus and we have done nothing extra to filter it.

Table 2 shows the experimental results on the cQA dataset. In this experiment, each sample is composed of one question and its following several candidate answers. We delete the ones with only one answer to confirm there are at least two candidate answers for each question. The candidate answers are rearranged by post time, so that the real answers do not always appear next to the questions. In this group of experiment, no hand-annotating work is needed because the real answers have been labeled by cQA users.

Method	**P@1 (%)**	**MRR (%)**
Nearest Answer	36.05	56.33
Cosine Similarity	44.05	62.84
HowNet	41.10	58.75
KL divergence	43.75	63.10
DBN (without FT)	**56.20**	**70.56**
DBN (with FT)	**58.15**	**72.74**

Table 2: Results on cQA Dataset

From Table 2 we observe that all the approaches perform much better on this dataset. We attribute the improvements to the high quality QA corpus Baidu Zhidao offers: the candidate answers tend to be more formal than the ones in the forums, with less noise information included. In addition, the "Nearest Answer" strategy has reached 36.05% in P@1 on this dataset, which indicates quite a number of askers receive the real answers at the first answer post. This result has supported the idea of introducing position features. What's more, if the best answer appear immediately, the asker tends to lock down the question thread, which helps to reduce the noise information in the cQA corpus.

Despite the baseline methods' performances have been improved, our approaches still outperform them, with a 32.0% improvement in P@1 and a 15.3% improvement in MRR at least. On the cQA dataset, our model shows better performance than the previous experiment, which is expected because the training set and the testing set come from the same corpus, and the DBN model is more adaptive to the cQA data.

We have observed that, from both of the two groups of experiments, fine-tuning is effective for enhancing the performance of our model. On the forum data, the results have been improved by 8.6% in P@1 and 4.0% in MRR, and the improvements are 3.5% and 3.1% individually.

6 Conclusions

In this paper, we have proposed a deep belief network based approach to model the semantic relevance for the question answering pairs in social community corpora.

The contributions of this paper can be summarized as follows: (1) The deep belief network we present shows good performance on modeling the QA pairs' semantic relevance using only word features. As a data driven approach, our model learns semantic knowledge from large amount of QA pairs to represent the semantic relevance between questions and their answers. (2) We have studied the textual similarity between the cQA and the forum datasets for QA pair extraction, and introduce a novel learning strategy to make our method show good performance on both cQA and forum datasets. The experimental results show that our method outperforms the traditional approaches on both the cQA and the forum corpora.

Our future work will be carried out along two directions. Firstly, we will further improve the performance of our method by adopting the non-textual features. Secondly, more research will be taken to put forward other architectures of the deep networks for QA detection.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. Special thanks to Deyuan Zhang, Bin Liu, Beidong Liu and Ke Sun for insightful suggestions. This work is supported by NSFC (60973076).

References

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer-finding. In *In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–199.

Delphine Bernhard and Iryna Gurevych. 2009. Combining lexical semantic resources with question & answer archives for translation-based answer finding. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 728–736, Suntec, Singapore, August. Association for Computational Linguistics.

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 467–474, New York, NY, USA. ACM.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. In *Proceedings of ACL-08: HLT*, pages 710–718, Columbus, Ohio, June. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *Proceedings of ACL-08: HLT*, pages 156–164, Columbus, Ohio, June. Association for Computational Linguistics.

Donghui Feng, Erin Shaw, Jihie Kim, and Eduard H. Hovy. 2006. An intelligent discussion-bot for answering student queries in threaded discussions. In Ccile Paris and Candace L. Sidner, editors, *IUI*, pages 171–177. ACM.

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Georey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14.

Liangjie Hong and Brian D. Davison. 2009. A classification-based approach to question answering in discussion boards. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 171–178, New York, NY, USA. ACM.

Jizhou Huang, Ming Zhou, and Dan Yang. 2007. Extracting chatbot knowledge from online discussion forums. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 423–428, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *CIKM '05*, pages 84–90, New York, NY, USA. ACM.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *SIGIR '06*, pages 228–235, New York, NY, USA. ACM.

Valentin Jijkoun and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the web. In *CIKM '05*, pages 76–83, New York, NY, USA. ACM.

Jung-Tae Lee, Sang-Bum Kim, Young-In Song, and Hae-Chang Rim. 2008. Bridging lexical gaps between queries and questions on large online q&a collections with compact translation models. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 410–418, Morristown, NJ, USA. Association for Computational Linguistics.

Fangtao Li, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering opinion questions with random walks on graphs. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 737–745, Suntec, Singapore, August. Association for Computational Linguistics.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, Prague, Czech Republic, June. Association for Computational Linguistics.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978.

Lokesh Shrestha and Kathleen McKeown. 2004. Detection of question-answer pairs in email conversations. In *Proceedings of Coling 2004*, pages 889–895, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-08: HLT*, pages 719–727, Columbus, Ohio, June. Association for Computational Linguistics.

Baoxun Wang, Bingquan Liu, Chengjie Sun, Xiaolong Wang, and Lin Sun. 2009. Extracting chinese question-answer pairs from online forums. In *SMC 2009: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2009.*, pages 1159–1164.

How Many Words is a Picture Worth? Automatic Caption Generation for News Images

Yansong Feng and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
Y.Feng-4@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we tackle the problem of automatic caption generation for news images. Our approach leverages the vast resource of pictures available on the web and the fact that many of them are captioned. Inspired by recent work in summarization, we propose *extractive* and *abstractive* caption generation models. They both operate over the output of a probabilistic image annotation model that preprocesses the pictures and suggests keywords to describe their content. Experimental results show that an abstractive model defined over phrases is superior to extractive methods.

1 Introduction

Recent years have witnessed an unprecedented growth in the amount of digital information available on the Internet. Flickr, one of the best known photo sharing websites, hosts more than three billion images, with approximately 2.5 million images being uploaded every day.[1] Many on-line news sites like CNN, Yahoo!, and BBC publish images with their stories and even provide photo feeds related to current events. Browsing and finding pictures in large-scale and heterogeneous collections is an important problem that has attracted much interest within information retrieval.

Many of the search engines deployed on the web retrieve images without analyzing their content, simply by matching user queries against collocated textual information. Examples include meta-data (e.g., the image's file name and format), user-annotated tags, captions, and generally text surrounding the image. As this limits the applicability of search engines (images that do not coincide with textual data cannot be retrieved), a great deal of work has focused on the development of methods that generate description words for a picture *automatically*. The literature is littered with various attempts to learn the associations between image features and words using supervised classification (Vailaya et al., 2001; Smeulders et al., 2000), instantiations of the noisy-channel model (Duygulu et al., 2002), latent variable models (Blei and Jordan, 2003; Barnard et al., 2002; Wang et al., 2009), and models inspired by information retrieval (Lavrenko et al., 2003; Feng et al., 2004).

In this paper we go one step further and generate captions for images rather than individual keywords. Although image indexing techniques based on keywords are popular and the method of choice for image retrieval engines, there are good reasons for using more linguistically meaningful descriptions. A list of keywords is often ambiguous. An image annotated with the words *blue*, *sky*, *car* could depict a blue car or a blue sky, whereas the caption "*car running under the blue sky*" would make the relations between the words explicit. Automatic caption generation could improve image retrieval by supporting longer and more targeted queries. It could also assist journalists in creating descriptions for the images associated with their articles. Beyond image retrieval, it could increase the accessibility of the web for visually impaired (blind and partially sighted) users who cannot access the content of many sites in the same ways as sighted users can (Ferres et al., 2006).

We explore the feasibility of automatic caption generation in the news domain, and create descriptions for images associated with on-line articles. Obtaining training data in this setting does not require expensive manual annotation as many articles are published together with captioned images. Inspired by recent work in summarization, we propose *extractive* and *abstractive* caption gen-

[1] http://www.techcrunch.com/2008/11/03/three-billion-photos-at-flickr/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1239–1249,
Uppsala, Sweden, 11-16 July 2010.

eration models. The backbone for both approaches is a probabilistic image annotation model that suggests keywords for an image. We can then simply identify (and rank) the sentences in the documents that share these keywords or create a new caption that is potentially more concise but also informative and fluent. Our abstractive model operates over image description keywords and document phrases. Their combination gives rise to many caption realizations which we select probabilistically by taking into account dependency and word order constraints. Experimental results show that the model's output compares favorably to handwritten captions and is often superior to extractive methods.

2 Related Work

Although image understanding is a popular topic within computer vision, relatively little work has focused on the interplay between visual and linguistic information. A handful of approaches generate image descriptions automatically following a two-stage architecture. The picture is first analyzed using image processing techniques into an abstract representation, which is then rendered into a natural language description with a text generation engine. A common theme across different models is domain specificity, the use of handlabeled data, and reliance on background ontological information.

For example, Héde et al. (2004) generate descriptions for images of objects shot in uniform background. Their system relies on a manually created database of objects indexed by an image signature (e.g., color and texture) and two keywords (the object's name and category). Images are first segmented into objects, their signature is retrieved from the database, and a description is generated using templates. Kojima et al. (2002, 2008) create descriptions for human activities in office scenes. They extract features of human motion and interleave them with a concept hierarchy of actions to create a case frame from which a natural language sentence is generated. Yao et al. (2009) present a general framework for generating text descriptions of image and video content based on image parsing. Specifically, images are hierarchically decomposed into their constituent visual patterns which are subsequently converted into a semantic representation using WordNet. The image parser is trained on a corpus, manually annotated with graphs representing image structure. A multi-sentence description is generated using a document planner and a surface realizer.

Within natural language processing most previous efforts have focused on generating captions to accompany complex graphical presentations (Mittal et al., 1998; Corio and Lapalme, 1999; Fasciano and Lapalme, 2000; Feiner and McKeown, 1990) or on using the captions accompanying information graphics to infer their intended message, e.g., the author's goal to convey ostensible increase or decrease of a quantity of interest (Elzer et al., 2005). Little emphasis is placed on image processing; it is assumed that the data used to create the graphics are available, and the goal is to enable users understand the information expressed in them.

The task of generating captions for news images is novel to our knowledge. Instead of relying on manual annotation or background ontological information we exploit a multimodal database of news articles, images, and their captions. The latter is admittedly noisy, yet can be easily obtained from on-line sources, and contains rich information about the entities and events depicted in the images and their relations. Similar to previous work, we also follow a two-stage approach. Using an image annotation model, we first describe the picture with keywords which are subsequently realized into a human readable sentence. The caption generation task bears some resemblance to headline generation (Dorr et al., 2003; Banko et al., 2000; Jin and Hauptmann, 2002) where the aim is to create a very short summary for a document. Importantly, we aim to create a caption that not only summarizes the document but is also a faithful to the image's content (i.e., the caption should also mention some of the objects or individuals depicted in the image). We therefore explore extractive and abstractive models that rely on visual information to drive the generation process. Our approach thus differs from most work in summarization which is solely text-based.

3 Problem Formulation

We formulate image caption generation as follows. Given an image I, and a related knowledge database κ, create a natural language description C which captures the main content of the image under κ. Specifically, in the news story scenario, we will generate a caption C for an image I and its accompanying document D. The training data thus consists of document-image-caption tu-

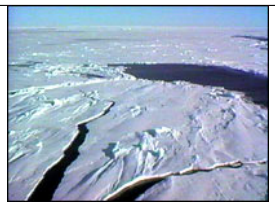
Thousands of Tongans have attended the funeral of King Taufa'ahau Tupou IV, who died last week at the age of 88. Representatives from 30 foreign countries watched as the king's coffin was carried by 1,000 men to the official royal burial ground.	 **King Tupou, who was 88, died a week ago.**	Contaminated Cadbury's chocolate was the most likely cause of an outbreak of salmonella poisoning, the Health Protection Agency has said. About 36 out of a total of 56 cases of the illness reported between March and July could be linked to the product.	 **Cadbury will increase its contamination testing levels.**
A Nasa satellite has documented startling changes in Arctic sea ice cover between 2004 and 2005. The extent of "perennial" ice declined by 14%, losing an area the size of Pakistan or Turkey. The last few decades have seen ice cover shrink by about 0.7% per year.	 **Satellite instruments can distinguish "old" Arctic ice from "new".**	A third of children in the UK use blogs and social network websites but two thirds of parents do not even know what they are, a survey suggests. The children's charity NCH said there was "an alarming gap" in technological knowledge between generations.	 **Children were found to be far more internet-wise than parents.**

Table 1: Each entry in the BBC News database contains a document an image, and its caption.

ples like the ones shown in Table 1. During testing, we are given a document and an associated image for which we must generate a caption.

Our experiments used the dataset created by Feng and Lapata (2008).[2] It contains 3,361 articles downloaded from the BBC News website[3] each of which is associated with a captioned news image. The latter is usually 203 pixels wide and 152 pixels high. The average caption length is 9.5 words, the average sentence length is 20.5 words, and the average document length 421.5 words. The caption vocabulary is 6,180 words and the document vocabulary is 26,795. The vocabulary shared between captions and documents is 5,921 words. The captions tend to use half as many words as the document sentences, and more than 50% of the time contain words that are not attested in the document (even though they may be attested in the collection).

Generating image captions is a challenging task even for humans, let alone computers. Journalists are given explicit instructions on how to write captions[4] and laypersons do not always agree on what a picture depicts (von Ahn and Dabbish, 2004). Along with the title, the lead, and section headings, captions are the most commonly read words in an article. A good caption must be succinct and informative, clearly identify the subject of the picture, establish the picture's relevance to the article, provide context for the picture, and ultimately draw the reader into the article. It is also worth noting that journalists often write their own captions rather than simply extract sentences from the document. In doing so they rely on general world knowledge but also expertise in current affairs that goes beyond what is described in the article or shown in the picture.

4 Image Annotation

As mentioned earlier, our approach relies on an image annotation model to provide description keywords for the picture. Our experiments made use of the probabilistic model presented in Feng and Lapata (2010). The latter is well-suited to our task as it has been developed with noisy, multimodal data sets in mind. The model is based on the assumption that images and their surrounding text are generated by mixtures of latent topics which are inferred from a concatenated representation of words and visual features.

Specifically, images are preprocessed so that they are represented by word-like units. Local image descriptors are computed using the Scale Invariant Feature Transform (SIFT) algorithm (Lowe, 1999). The general idea behind the algorithm is to first sample an image with the difference-of-Gaussians point detector at different

[2] Available from http://homepages.inf.ed.ac.uk/s677528/data/

[3] http://news.bbc.co.uk/

[4] See http://www.theslot.com/captions.html and http://www.thenewsmanual.net/ for tips on how to write good captions.

scales and locations. Importantly, this detector is, to some extent, invariant to translation, scale, rotation and illumination changes. Each detected region is represented with a SIFT descriptor which is a histogram of edge directions at different locations. Subsequently SIFT descriptors are quantized into a discrete set of visual terms via a clustering algorithm such as K-means.

The model thus works with a bag-of-words representation and treats each article-image-caption tuple as a single document d_{Mix} consisting of textual and visual words. Latent Dirichlet Allocation (LDA, Blei et al. 2003) is used to infer the latent topics assumed to have generated d_{Mix}. The basic idea underlying LDA, and topic models in general, is that each document is composed of a probability distribution over topics, where each topic represents a probability distribution over words. The document-topic and topic-word distributions are learned automatically from the data and provide information about the semantic themes covered in each document and the words associated with each semantic theme. The image annotation model takes the topic distributions into account when finding the most likely keywords for an image and its associated document.

More formally, given an image-caption-document tuple (I,C,D) the model finds the subset of keywords W_I ($W_I \subseteq W$) which appropriately describe I. Assuming that keywords are conditionally independent, and I, D are represented jointly by d_{Mix}, the model estimates:

$$\begin{aligned} W_I^* &\approx \arg\max_{W_t} \prod_{w_t \in W_t} P(w_t|d_{Mix}) \qquad (1) \\ &= \arg\max_{W_t} \prod_{w_t \in W_t} \sum_{k=1}^{K} P(w_t|z_k)P(z_k|d_{Mix}) \end{aligned}$$

W_t denotes a set of description keywords (the subscript t is used to discriminate from the visual words which are not part of the model's output), K the number of topics, $P(w_t|z_k)$ the multimodal word distributions over topics, and $P(z_k|d_{Mix})$ the estimated posterior of the topic proportions over documents. Given an unseen image-document pair and trained multimodal word distributions over topics, it is possible to infer the posterior of topic proportions over the new data by maximizing the likelihood. The model delivers a ranked list of textual words w_t, the n-best of which are used as annotations for image I.

It is important to note that the caption generation models we propose are not especially tied to the above annotation model. Any probabilistic model with broadly similar properties could serve our purpose. Examples include PLSA-based approaches to image annotation (e.g., Monay and Gatica-Perez 2007) and correspondence LDA (Blei and Jordan, 2003).

5 Extractive Caption Generation

Much work in summarization to date focuses on sentence extraction where a summary is created simply by identifying and subsequently concatenating the most important sentences in a document. Without a great deal of linguistic analysis, it is possible to create summaries for a wide range of documents, independently of style, text type, and subject matter. For our caption generation task, we need only extract a single sentence. And our guiding hypothesis is that this sentence must be maximally similar to the description keywords generated by the annotation model. We discuss below different ways of operationalizing similarity.

Word Overlap Perhaps the simplest way of measuring the similarity between image keywords and document sentences is word overlap:

$$Overlap(W_I,S_d) = \frac{|W_I \cap S_d|}{|W_I \cup S_d|} \qquad (2)$$

where W_I is the set of keywords and S_d a sentence in the document. The caption is then the sentence that has the highest overlap with the keywords.

Cosine Similarity Word overlap is admittedly a naive measure of similarity, based on lexical identity. We can overcome this by representing keywords and sentences in vector space (Salton and McGill, 1983). The latter is a word-sentence co-occurrence matrix where each row represents a word, each column a sentence, and each entry the frequency with which the word appeared within the sentence. More precisely matrix cells are weighted by their tf-idf values. The similarity of the vectors representing the keywords $\overrightarrow{W_I}$ and document sentence $\overrightarrow{S_d}$ can be quantified by measuring the cosine of their angle:

$$sim(\overrightarrow{W_I},\overrightarrow{S_d}) = \frac{\overrightarrow{W_I} \cdot \overrightarrow{S_d}}{|W_I||\overrightarrow{S_d}|} \qquad (3)$$

Probabilistic Similarity Recall that the backbone of our image annotation model is a topic model with images and documents represented as a probability distribution over latent topics. Under this framework, the similarity between an im-

age and a sentence can be broadly measured by the extent to which they share the same topic distributions (Steyvers and Griffiths, 2007). For example, we may use the KL divergence to measure the difference between the distributions p and q:

$$D(p,q) = \sum_{j=1}^{K} p_j \log_2 \frac{p_j}{q_j} \quad (4)$$

where p and q are shorthand for the image topic distribution $P_{d_{Mix}}$ and sentence topic distribution P_{S_d}, respectively. When doing inference on the document sentence, we also take its neighboring sentences into account to avoid estimating inaccurate topic proportions on short sentences.

The KL divergence is asymmetric and in many applications, it is preferable to apply a symmetric measure such as the Jensen Shannon (JS) divergence. The latter measures the "distance" between p and q through $\frac{(p+q)}{2}$, the average of p and q:

$$JS(p,q) = \frac{1}{2}\left[D(p, \frac{(p+q)}{2}) + D(q, \frac{(p+q)}{2})\right] \quad (5)$$

6 Abstractive Caption Generation

Although extractive methods yield grammatical captions and require relatively little linguistic analysis, there are a few caveats to consider. Firstly, there is often no single sentence in the document that uniquely describes the image's content. In most cases the keywords are found in the document but interspersed across multiple sentences. Secondly, the selected sentences make for long captions (sometimes longer than the average document sentence), are not concise and overall not as catchy as human-written captions. For these reasons we turn to abstractive caption generation and present models based on single words but also phrases.

Word-based Model Our first abstractive model builds on and extends a well-known probabilistic model of headline generation (Banko et al., 2000). The task is related to caption generation, the aim is to create a short, title-like headline for a given document, without however taking visual information into account. Like captions, headlines have to be catchy to attract the reader's attention.

Banko et al. (2000) propose a bag-of-words model for headline generation. It consists of content selection and surface realization components. Content selection is modeled as the probability of a word appearing in the headline given the same word appearing in the corresponding document and is independent from other words in the headline. The likelihood of different surface realizations is estimated using a bigram model. They also take the distribution of the length of the headlines into account in an attempt to bias the model towards generating concise output:

$$\begin{aligned} P(w_1, w_2, ..., w_n) = & \prod_{i=1}^{n} P(w_i \in H | w_i \in D) \\ & \cdot P(len(H) = n) \\ & \cdot \prod_{i=2}^{n} P(w_i | w_{i-1}) \end{aligned} \quad (6)$$

where w_i is a word that may appear in headline H, D the document being summarized, and $P(len(H) = n)$ a headline length distribution model.

The above model can be easily adapted to the caption generation task. Content selection is now the probability of a word appearing in the caption given the image and its associated document which we obtain from the output of our image annotation model (see Section 4). In addition we replace the bigram surface realizer with a trigram:

$$\begin{aligned} P(w_1, w_2, ..., w_n) = & \prod_{i=1}^{n} P(w_i \in C | I, D) \\ & \cdot P(len(C) = n) \\ & \cdot \prod_{i=3}^{n} P(w_i | w_{i-1}, w_{i-2}) \end{aligned} \quad (7)$$

where C is the caption, I the image, D the accompanying document, and $P(w_i \in C | I, D)$ the image annotation probability.

Despite its simplicity, the caption generation model in (7) has a major drawback. The content selection component will naturally tend to ignore function words, as they are not descriptive of the image's content. This will seriously impact the grammaticality of the generated captions, as there will be no appropriate function words to glue the content words together. One way to remedy this is to revert to a content selection model that ignores the image and simply estimates the probability of a word appearing in the caption given the same word appearing in the document. At the same time we modify our surface realization component so that it takes note of the image annotation probabilities. Specifically, we use an *adaptive* language model (Kneser et al., 1997) that modifies an

n-gram model with local unigram probabilities:

$$P(w_1,w_2,...,w_n) = \prod_{i=1}^{n} P(w_i \in C | w_i \in D) \cdot P(len(C)=n) \cdot \prod_{i=3}^{n} P_{adap}(w_i|w_{i-1},w_{i-2}) \quad (8)$$

where $P(w_i \in C|w_i \in D)$ is the probability of w_i appearing in the caption given that it appears in the document D, and $P_{adap}(w_i|w_{i-1},w_{i-2})$ the language model adapted with probabilities from our image annotation model:

$$P_{adap}(w|h) = \frac{\alpha(w)}{z(h)} P_{back}(w|h) \quad (9)$$

$$\alpha(w) \approx (\frac{P_{adap}(w)}{P_{back}(w)})^{\beta} \quad (10)$$

$$z(h) = \sum_{w} \alpha(w) \cdot P_{back}(w|h) \quad (11)$$

where $P_{back}(w|h)$ is the probability of w given the history h of preceding words (i.e., the original trigram model), $P_{adap}(w)$ the probability of w according to the image annotation model, $P_{back}(w)$ the probability of w according to the original model, and β a scaling parameter.

Phrase-based Model The model outlined in equation (8) will generate captions with function words. However, there is no guarantee that these will be compatible with their surrounding context or that the caption will be globally coherent beyond the trigram horizon. To avoid these problems, we turn our attention to phrases which are naturally associated with function words and can potentially capture long-range dependencies.

Specifically, we obtain phrases from the output of a dependency parser. A phrase is simply a head and its dependents with the exception of verbs, where we record only the head (otherwise, an entire sentence could be a phrase). For example, from the first sentence in Table 1 (first row, left document) we would extract the phrases: *thousands of Tongans*, *attended*, *the funeral*, *King Taufa'ahau Tupou IV*, *last week*, *at the age*, *died*, and so on. We only consider dependencies whose heads are nouns, verbs, and prepositions, as these constitute 80% of all dependencies attested in our caption data. We define a bag-of-phrases model for caption generation by modifying the content selection and caption length components in equation (8) as follows:

$$P(\rho_1,\rho_2,...,\rho_m) \approx \prod_{j=1}^{m} P(\rho_j \in C|\rho_j \in D) \cdot P(len(C) = \sum_{j=1}^{m} len(\rho_j)) \cdot \prod_{i=3}^{\sum_{j=1}^{m} len(\rho_j)} P_{adap}(w_i|w_{i-1},w_{i-2}) \quad (12)$$

Here, $P(\rho_j \in C|\rho_j \in D)$ models the probability of phrase ρ_j appearing in the caption given that it also appears in the document and is estimated as:

$$P(\rho_j \in C|\rho_j \in D) = \prod_{w_j \in \rho_j} P(w_j \in C|w_j \in D) \quad (13)$$

where w_j is a word in the phrase ρ_j.

One problem with the models discussed thus far is that words or phrases are independent of each other. It is up to the trigram model to enforce coarse ordering constraints. These may be sufficient when considering isolated words, but phrases are longer and their combinations are subject to structural constraints that are not captured by sequence models. We therefore attempt to take phrase *attachment* constraints into account by estimating the probability of phrase ρ_j attaching to the right of phrase ρ_i as:

$$P(\rho_j|\rho_i) = \sum_{w_i \in \rho_i} \sum_{w_j \in \rho_j} p(w_j|w_i) = \frac{1}{2} \sum_{w_i \in \rho_i} \sum_{w_j \in \rho_j} \{\frac{f(w_i,w_j)}{f(w_i,-)} + \frac{f(w_i,w_j)}{f(-,w_j)}\} \quad (14)$$

where $p(w_j|w_i)$ is the probability of a phrase containing word w_j appearing to the right of a phrase containing word w_i, $f(w_i,w_j)$ indicates the number of times w_i and w_j are adjacent, $f(w_i,-)$ is the number of times w_i appears on the left of any phrase, and $f(-,w_i)$ the number of times it appears on the right.[5]

After integrating the attachment probabilities into equation (12), the caption generation model becomes:

$$P(\rho_1,\rho_2,...,\rho_m) \approx \prod_{j=1}^{m} P(\rho_j \in C|\rho_j \in D) \cdot \prod_{j=2}^{m} P(\rho_j|\rho_{j-1}) \cdot P(len(C) = \textstyle\sum_{j=1}^{m} len(\rho_j)) \cdot \prod_{i=3}^{\sum_{j=1}^{m} len(\rho_j)} P_{adap}(w_i|w_{i-1},w_{i-2}) \quad (15)$$

[5] Equation (14) is smoothed to avoid zero probabilities.

On the one hand, the model in equation (15) takes long distance dependency constraints into account, and has some notion of syntactic structure through the use of attachment probabilities. On the other hand, it has a primitive notion of caption length estimated by $P(len(C) = \sum_{j=1}^{m} len(\rho_j))$ and will therefore generate captions of the same (phrase) length. Ideally, we would like the model to vary the length of its output depending on the chosen context. However, we leave this to future work.

Search To generate a caption it is necessary to find the sequence of words that maximizes $P(w_1, w_2, ..., w_n)$ for the word-based model (equation (8)) and $P(\rho_1, \rho_2, ..., \rho_m)$ for the phrase-based model (equation (15)). We rewrite both probabilities as the weighted sum of their log form components and use beam search to find a near-optimal sequence. Note that we can make search more efficient by reducing the size of the document D. Using one of the models from Section 5, we may rank its sentences in terms of their relevance to the image keywords and consider only the n-best ones. Alternatively, we could consider the single most relevant sentence together with its surrounding context under the assumption that neighboring sentences are about the same or similar topics.

7 Experimental Setup

In this section we discuss our experimental design for assessing the performance of the caption generation models presented above. We give details on our training procedure, parameter estimation, and present the baseline methods used for comparison with our models.

Data All our experiments were conducted on the corpus created by Feng and Lapata (2008), following their original partition of the data (2,881 image-caption-document tuples for training, 240 tuples for development and 240 for testing). Documents and captions were parsed with the Stanford parser (Klein and Manning, 2003) in order to obtain dependencies for the phrase-based abstractive model.

Model Parameters For the image annotation model we extracted 150 (on average) SIFT features which were quantized into 750 visual terms. The underlying topic model was trained with 1,000 topics using only content words (i.e., nouns, verbs, and adjectives) that appeared no less than five times in the corpus. For all models discussed here (extractive and abstractive) we report results with the 15 best annotation keywords. For the abstractive models, we used a trigram model trained with the SRI toolkit on a newswire corpus consisting of BBC and Yahoo! news documents (6.9 M words). The attachment probabilities (see equation (14)) were estimated from the same corpus. We tuned the caption length parameter on the development set using a range of $[5, 14]$ tokens for the word-based model and $[2, 5]$ phrases for the phrase-based model. Following Banko et al. (2000), we approximated the length distribution with a Gaussian. The scaling parameter β for the adaptive language model was also tuned on the development set using a range of [0.5,0.9]. We report results with β set to 0.5. For the abstractive models the beam size was set to 500 (with at least 50 states for the word-based model). For the phrase-based model, we also experimented with reducing the search scope, either by considering only the n most similar sentences to the keywords (range $[2, 10]$), or simply the single most similar sentence and its neighbors (range $[2, 5]$). The former method delivered better results with 10 sentences (and the KL divergence similarity function).

Evaluation We evaluated the performance of our models automatically, and also by eliciting human judgments. Our automatic evaluation was based on Translation Edit Rate (TER, Snover et al. 2006), a measure commonly used to evaluate the quality of machine translation output. TER is defined as the minimum number of edits a human would have to perform to change the system output so that it exactly matches a reference translation. In our case, the original captions written by the BBC journalists were used as reference:

$$\text{TER}(E, E_r) = \frac{\text{Ins} + \text{Del} + \text{Sub} + \text{Shft}}{N_r} \qquad (16)$$

where E is the hypothetical system output, E_r the reference caption, and N_r the reference length. The number of possible edits include insertions (Ins), deletions (Del), substitutions (Sub) and shifts (Shft). TER is similar to word error rate, the only difference being that it allows shifts. A shift moves a contiguous sequence to a different location within the the same system output and is counted as a single edit. The perfect TER score is 0, however note that it can be higher than 1 due to insertions. The minimum translation edit align-

Model	TER	AvgLen
Lead sentence	$2.12^{\dagger}$	21.0
Word Overlap	$2.46^{*\dagger}$	24.3
Cosine	$2.26^{\dagger}$	22.0
KL Divergence	$1.77^{*\dagger}$	18.4
JS Divergence	$1.77^{*\dagger}$	18.6
Abstract Words	$1.11^{*\dagger}$	10.0
Abstract Phrases	$1.06^{*\dagger}$	10.1

Table 2: TER results for extractive, abstractive models, and lead sentence baseline; *: sig. different from lead sentence; †: sig. different from KL and JS divergence.

ment is usually found through beam search. We used TER to compare the output of our extractive and abstractive models and also for parameter tuning (see the discussion above).

In our human evaluation study participants were presented with a document, an associated image, and its caption, and asked to rate the latter on two dimensions: grammaticality (is the sentence fluent or word salad?) and relevance (does it describe succinctly the content of the image and document?). We used a 1–7 rating scale, participants were encouraged to give high ratings to captions that were grammatical and appropriate descriptions of the image given the accompanying document. We randomly selected 12 document-image pairs from the test set and generated captions for them using the best extractive system, and two abstractive systems (word-based and phrase-based). We also included the original human-authored caption as an upper bound. We collected ratings from 23 unpaid volunteers, all self reported native English speakers. The study was conducted over the Internet.

8 Results

Table 2 reports our results on the test set using TER. We compare four extractive models based on word overlap, cosine similarity, and two probabilistic similarity measures, namely KL and JS divergence and two abstractive models based on words (see equation (8)) and phrases (see equation (15)). We also include a simple baseline that selects the first document sentence as a caption and show the average caption length (AvgLen) for each model. We examined whether performance differences among models are statistically significant, using the Wilcoxon test.

Model	Grammaticality	Relevance
KL Divergence	$6.42^{*\dagger}$	$4.10^{*\dagger}$
Abstract Words	$2.08^{\dagger}$	$3.20^{\dagger}$
Abstract Phrases	4.80^{*}	4.96^{*}
Gold Standard	$6.39^{*\dagger}$	5.55^{*}

Table 3: Mean ratings on caption output elicited by humans; *: sig. different from word-based abstractive system; †: sig. different from phrase-based abstractive system.

As can be seen the probabilistic models (KL and JS divergence) outperform word overlap and cosine similarity (all differences are statistically significant, $p < 0.01$).[6] They make use of the same topic model as the image annotation model, and are thus able to select sentences that cover common content. They are also significantly better than the lead sentence which is a competitive baseline. It is well known that news articles are written so that the lead contains the most important information in a story.[7] This is an encouraging result as it highlights the importance of the visual information for the caption generation task. In general, word overlap is the worst performing model which is not unexpected as it does not take any lexical variation into account. Cosine is slightly better but not significantly different from the lead sentence. The abstractive models obtain the best TER scores overall, however they generate shorter captions in comparison to the other models (closer to the length of the gold standard) and as a result TER treats them favorably, simply because the number of edits is less. For this reason we turn to the results of our judgment elicitation study which assesses in more detail the quality of the generated captions.

Recall that participants judge the system output on two dimensions, grammaticality and relevance. Table 3 reports mean ratings for the output of the extractive system (based on the KL divergence), the two abstractive systems, and the human-authored gold standard caption. We performed an Analysis of Variance (ANOVA) to examine the effect of system type on the generation task. Post-hot Tukey tests were carried out on the mean of the ratings shown in Table 3 (for grammaticality and relevance).

[6] We also note that mean length differences are not significant among these models.

[7] As a rule of thumb the lead should answer most or all of the five W's (who, what, when, where, why).

G:	King Tupou, who was 88, died a week ago.
KL:	Last year, thousands of Tongans took part in unprecedented demonstrations to demand greater democracy and public ownership of key national assets.
A_W:	King Toupou IV died at the age of Tongans last week.
A_P:	King Toupou IV died at the age of 88 last week.
G:	Cadbury will increase its contamination testing levels.
KL:	Contaminated Cadbury's chocolate was the most likely cause of an outbreak of salmonella poisoning, the Health Protection Agency has said.
A_W:	Purely dairy milk buttons Easter had agreed to work has caused.
A_P:	The 105g dairy milk buttons Easter egg affected by the recall.
G:	Satellite instruments can distinguish "old" Arctic ice from "new".
KL:	So a planet with less ice warms faster, potentially turning the projected impacts of global warming into reality sooner than anticipated.
A_W:	Dr less winds through ice cover all over long time when.
A_P:	The area of the Arctic covered in Arctic sea ice cover.
G:	Children were found to be far more internet-wise than parents.
KL:	That's where parents come in.
A_W:	The survey found a third of children are about mobile phones.
A_P:	The survey found a third of children in the driving seat.

Table 4: Captions written by humans (G) and generated by extractive (KL), word-based abstractive (A_W), and phrase-based extractive (A_P systems).

The word-based system yields the least grammatical output. It is significantly worse than the phrase-based abstractive system ($\alpha < 0.01$), the extractive system ($\alpha < 0.01$), and the gold standard ($\alpha < 0.01$). Unsurprisingly, the phrase-based system is significantly less grammatical than the gold standard and the extractive system, whereas the latter is perceived as equally grammatical as the gold standard (the difference in the means is not significant). With regard to relevance, the word-based system is significantly worse than the phrase-based system, the extractive system, and the gold-standard. Interestingly, the phrase-based system performs on the same level with the human gold standard (the difference in the means is not significant) and significantly better than the extractive system. Overall, the captions generated by the phrase-based system, capture the same content as the human-authored captions, even though they tend to be less grammatical. Examples of system output for the image-document pairs shown in Table 1 are given in Table 4 (the first row corresponds to the left picture (top row) in Table 1, the second row to the right picture, and so on).

9 Conclusions

We have presented extractive and abstractive models that generate image captions for news articles. A key aspect of our approach is to allow both the visual and textual modalities to influence the generation task. This is achieved through an image annotation model that characterizes pictures in terms of description keywords that are subsequently used to guide the caption generation process. Our results show that the visual information plays an important role in content selection. Simply extracting a sentence from the document often yields an inferior caption. Our experiments also show that a probabilistic abstractive model defined over phrases yields promising results. It generates captions that are more grammatical than a closely related word-based system and manages to capture the gist of the image (and document) as well as the captions written by journalists.

Future extensions are many and varied. Rather than adopting a two-stage approach, where the image processing and caption generation are carried out sequentially, a more general model should integrate the two steps in a unified framework. Indeed, an avenue for future work would be to define a phrase-based model for both image annotation and caption generation. We also believe that our approach would benefit from more detailed linguistic and non-linguistic information. For instance, we could experiment with features related to document structure such as titles, headings, and sections of articles and also exploit syntactic information more directly. The latter is currently used in the phrase-based model by taking attachment probabilities into account. We could, however, improve grammaticality more globally by generating a well-formed tree (or dependency graph).

References

Banko, Michel, Vibhu O. Mittal, and Micheael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Hong Kong, pages 318–325.

Barnard, Kobus, Pinar Duygulu, David Forsyth, Nando de Freitas, David Blei, and Michael Jordan. 2002. Matching words and pictures. *Journal of Machine Learning Research* 3:1107–1135.

Blei, David and Michael Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th An-*

nual International ACM SIGIR Conference on Research and Development in Information Retrieval. Toronto, ON, pages 127–134.

Blei, David, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Corio, Marc and Guy Lapalme. 1999. Generation of texts for information graphics. In *Proceedings of the 7th European Workshop on Natural Language Generation*. Toulouse, France, pages 49–58.

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization*. Edmonton, Canada, pages 1–8.

Duygulu, Pinar, Kobus Barnard, Nando de Freitas, and David Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision*. Copenhagen, Denmark, pages 97–112.

Elzer, Stephanie, Sandra Carberry, Ingrid Zukerman, Daniel Chester, Nancy Green, , and Seniz Demir. 2005. A probabilistic framework for recognizing intention in information graphics. In *Proceedings of the 19th International Conference on Artificial Intelligence*. Edinburgh, Scotland, pages 1042–1047.

Fasciano, Massimo and Guy Lapalme. 2000. Intentions in the coordinated generation of graphics and text from tabular data. *Knowledge Information Systems* 2(3):310–339.

Feiner, Steven and Kathleen McKeown. 1990. Coordinating text and graphics in explanation generation. In *Proceedings of National Conference on Artificial Intelligence*. Boston, MA, pages 442–449.

Feng, Shaolei Feng, Victor Lavrenko, and R Manmatha. 2004. Multiple Bernoulli relevance models for image and video annotation. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. Washington, DC, pages 1002–1009.

Feng, Yansong and Mirella Lapata. 2008. Automatic image annotation using auxiliary text information. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies*. Columbus, OH, pages 272–280.

Feng, Yansong and Mirella Lapata. 2010. Topic models for image annotation and text illustration. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, LA.

Ferres, Leo, Avi Parush, Shelley Roberts, and Gitte Lindgaard. 2006. Helping people with visual impairments gain access to graphical information through natural language: The *graph* system. In *Proceedings of 11th International Conference on Computers Helping People with Special Needs*. Linz, Austria, pages 1122–1130.

Héde, Patrick, Pierre Allain Moëllic, Joël Bourgeoys, Magali Joint, and Corinne Thomas. 2004. Automatic generation of natural language descriptions for images. In *Proceedings of Computer-Assisted Information Retrieval (Recherche d'Information et ses Applications Ordinateur) (RIAO)*. Avignon, France.

Jin, Rong and Alexander G. Hauptmann. 2002. A new probabilistic model for title generation. In *Proceedings of the 19th International Conference on Computational linguistics*. Taipei, Taiwan, pages 1–7.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics*. Sapporo, Japan, pages 423–430.

Kneser, Reinhard, Jochen Peters, and Dietrich Klakow. 1997. Language model adaptation using dynamic marginals. In *Proceedings of 5th European Conference on Speech Communication and Technology*. Rhodes, Greece, volume 4, pages 1971–1974.

Kojima, Atsuhiro, Mamoru Takaya, Shigeki Aoki, Takao Miyamoto, and Kunio Fukunaga. 2008. Recognition and textual description of human activities by mobile robot. In *Proceedings of the 3rd International Conference on Innovative Computing Information and Control*. IEEE Computer Society, Washington, DC, pages 53–56.

Kojima, Atsuhiro, Takeshi Tamura, and Kunio Fukunaga. 2002. Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision* 50(2):171–184.

Lavrenko, Victor, R. Manmatha, and Jiwoon Jeon. 2003. A model for learning the semantics of

pictures. In *Proceedings of the 16th Conference on Advances in Neural Information Processing Systems*. Vancouver, BC.

Lowe, David G. 1999. Object recognition from local scale-invariant features. In *Proceedings of International Conference on Computer Vision*. IEEE Computer Society, pages 1150–1157.

Mittal, Vibhu O., Johanna D. Moore, Giuseppe Carenini, and Steven Roth. 1998. Describing complex charts in natural language: A caption generation system. *Computational Linguistics* 24:431–468.

Monay, Florent and Daniel Gatica-Perez. 2007. Modeling semantic aspects for cross-media image indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(10):1802–1817.

Salton, Gerard and M.J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Smeulders, Arnols W.M., Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(12):1349–1380.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*. Cambridge, pages 223–231.

Steyvers, Mark and Tom Griffiths. 2007. Probabilistic topic models. In T. Landauer, D. McNamara, S Dennis, and W Kintsch, editors, *A Handbook of Latent Semantic Analysis*, Psychology Press.

Vailaya, Aditya, Mário A. T. Figueiredo, Anil K. Jain, and Hong-Jiang Zhang. 2001. Image classification for content-based indexing. *IEEE Transactions on Image Processing* 10:117–130.

von Ahn, Luis and Laura Dabbish. 2004. Labeling images with a computer game. In *ACM Conference on Human Factors in Computing Systems*. New York, NY, pages 319–326.

Wang, Chong, David Blei, and Li Fei-Fei. 2009. Simultaneous image classification and annotation. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. Miami, FL, pages 1903–1910.

Yao, Benjamin, Xiong Yang, Liang Lin, Mun Wai Lee, and Song chun Zhu. 2009. I2t: Image parsing to text description. *Proceedings of IEEE (invited for the special issue on Internet Vision)* .

Generating image descriptions using dependency relational patterns

Ahmet Aker
University of Sheffield
a.aker@dcs.shef.ac.uk

Robert Gaizauskas
University of Sheffield
r.gaizauskas@dcs.shef.ac.uk

Abstract

This paper presents a novel approach to automatic captioning of geo-tagged images by summarizing multiple web-documents that contain information related to an image's location. The summarizer is biased by dependency pattern models towards sentences which contain features typically provided for different scene types such as those of churches, bridges, etc. Our results show that summaries biased by dependency pattern models lead to significantly higher ROUGE scores than both n-gram language models reported in previous work and also Wikipedia baseline summaries. Summaries generated using dependency patterns also lead to more readable summaries than those generated without dependency patterns.

1 Introduction

The number of images tagged with location information on the web is growing rapidly, facilitated by the availability of GPS (Global Position System) equipped cameras and phones, as well as by the widespread use of online social sites. The majority of these images are indexed with GPS coordinates (latitude and longitude) only and/or have minimal captions. This typically small amount of textual information associated with the image is of limited usefulness for image indexing, organization and search. Therefore methods which could automatically supplement the information available for image indexing and lead to improved image retrieval would be extremely useful.

Following the general approach proposed by Aker and Gaizauskas (2009), in this paper we describe a method for automatic image captioning or caption enhancement starting with only a scene or subject type and a set of place names pertaining to an image – for example $\langle$church, $\{$St. Paul's,London$\}\rangle$. Scene type and place names can be obtained automatically given GPS coordinates and compass information using techniques such as those described in Xin et al. (2010) – that task is not the focus of this paper.

Our method applies only to images of static features of the built or natural landscape, i.e. objects with persistent geo-coordinates, such as buildings and mountains, and not to images of objects which move about in such landscapes, e.g. people, cars, clouds, etc. However, our technique is suitable not only for image captioning but in any application context that requires summary descriptions of instances of object classes, where the instance is to be characterized in terms of the features typically mentioned in describing members of the class.

Aker and Gaizauskas (2009) have argued that humans appear to have a conceptual model of what is salient regarding a certain object type (e.g. church, bridge, etc.) and that this model informs their choice of what to say when describing an instance of this type. They also experimented with representing such conceptual models using n-gram language models derived from corpora consisting of collections of descriptions of instances of specific object types (e.g. a corpus of descriptions of churches, a corpus of bridge descriptions, and so on) and reported results showing that incorporating such n-gram language models as a feature in a feature-based extractive summarizer improves the quality of automatically generated summaries.

The main weakness of n-gram language models is that they only capture very local information about short term sequences and cannot model long distance dependencies between terms. For example one common and important feature of object descriptions is the simple specification of the object type, e.g. the information that the object *London Bridge* is a *bridge* or that the *Rhine* is a *river*. If this information is expressed as in the first line of Table 1, n-gram language models are likely to

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1250–1258,
Uppsala, Sweden, 11-16 July 2010.

Table 1: Example of sentences which express the type of an object.

London Bridge is a **bridge**...
The **Rhine** (German: Rhein; Dutch: Rijn; French: Rhin; Romansh: Rain; Italian: Reno; Latin: Rhenus West Frisian Ryn) is one of the longest and most important **rivers** in Europe...

reflect it, since one would expect the tri-gram *is a bridge* to occur with high frequency in a corpus of bridge descriptions. However, if the type predication occurs with less commonly seen local context, as is the case for the object *Rhine* in the second row of Table 1 – *most important rivers* – n-gram language models may well be unable to identify it.

Intuitively, what is important in both these cases is that there is a predication whose subject is the object instance of interest and the head of whose complement is the object type: *London Bridge ... is ... bridge* and *Rhine ... is ... river*. Sentences matching such patterns are likely to be important ones to include in a summary. This intuition suggests that rather than representing object type conceptual models via corpus-derived language models as do Aker and Gaizauskas (2009), we do so instead using *corpus-derived dependency patterns*.

We pursue this idea in this paper, our hypothesis being that information that is important for describing objects of a given type will frequently be realized linguistically via expressions with the same dependency structure. We explore this hypothesis by developing a method for deriving common dependency patterns from object type corpora (Section 2) and then incorporating these patterns into an extractive summarization system (Section 3). In Section 4 we evaluate the approach both by scoring against model summaries and via a readability assessment. Since our work aims to extend the work of Aker and Gaizauskas (2009) we reproduce their experiments with n-gram language models in the current setting so as to permit accurate comparison.

Multi-document summarizers face the problem of avoiding redundancy: often, important information which must be included in the summary is repeated several times across the document set, but must be included in the summary only once. We can use the dependency pattern approach to address this problem in a novel way. The common approach to avoiding redundancy is to use a text similarity measure to block the addition of a further sentence to the summary if it is too similar to one already included. Instead, since specific dependency patterns express specific types of information we can group the patterns into groups expressing the same type of information and then, during sentence selection, ensure that sentences matching patterns from different groups are selected in order to guarantee broad, non-redundant coverage of information relevant for inclusion in the summary. We report work experimenting with this idea too.

Table 2: Object types and the number of articles in each object type corpus. Object types which are bold are covered by the evaluation image set.

village 39970, school 15794, city 14233, organization 9393, **university** 7101, **area** 6934, **district** 6565, airport 6493, **island** 6400, **railway station** 5905, **river** 5851, company 5734, **mountain** 5290, **park** 3754, **college** 3749, **stadium** 3665, **lake** 3649, **road** 3421, country 3186, **church** 3005, way 2508, **museum** 2320, **railway** 2093, **house** 2018, arena 1829, field 1731, club 1708, shopping centre 1509, highway 1464, **bridge** 1383, **street** 1352, **theatre** 1330, bank 1310, property 1261, **hill** 1072, **castle** 1022, forest 995, court 949, hospital 937, peak 906, bay 899, **skyscraper** 843, **valley** 763, **hotel** 741, **garden** 739, **building** 722, market 712, **monument** 679, port 651, sea 645, **temple** 625, **beach** 614, **square** 605, store 547, campus 525, **palace** 516, **tower** 496, cemetery 457, **volcano** 426, **cathedral** 402, **glacier** 392, **residence** 371, dam 363, **waterfall** 355, **gallery** 349, **prison** 348, **cave** 341, **canal** 332, restaurant 329, path 312, observatory 303, **zoo** 302, coast 298, **statue** 283, **venue** 269, **parliament** 258, shrine 256, desert 248, synagogue 236, bar 229, **ski resort** 227, arch 223, landscape 220, **avenue** 202, casino 179, farm 179, seaside 173, waterway 167, tunnel 167, ruin 166, **chapel** 165, **observation wheel** 158, **basilica** 157, woodland 154, wetland 151, cinema 144, **gate** 142, **aquarium** 136, entrance 136, **opera house** 134, **spa** 125, shop 124, **abbey** 108, **boulevard** 108, **pub** 92, bookstore 76, **mosque** 56

2 Representing conceptual models

2.1 Object type corpora

We derive n-gram language and dependency pattern models using object type corpora made available to us by Aker and Gaizauskas. Aker and Gaizauskas (2009) define an object type corpus as a collection of texts about a specific static object type such as *church, bridge, etc.* Objects can be named locations such as *Eiffel Tower*. To refer to such names they use the term *toponym*. To build such object type corpora the authors categorized Wikipedia articles places by object type. The object type of each article was identified automatically by running *Is-A* patterns over the first five sentences of the article. The authors report 91% accuracy for their categorization process. The most populated of the categories identified (in total 107 containing articles about places around the world) are shown in Table 2.

2.2 N-gram language models

Aker and Gaizauskas (2009) experimented with uni-gram and bi-gram language models to capture the features commonly used when describing an object type and used these to bias the sentence selection of the summarizer towards the sentences that contain these features. As in Song and Croft (1999) they used their language models in a gener-

ative way, i.e. they calculate the probability that a sentence is generated based on a n-gram language model. They showed that summarizer biased with bi-gram language models produced better results than those biased with uni-gram models. We replicate the experiments of Aker and Gaizauskas and generate a bi-gram language model for each object type corpus. In later sections we use *LM* to refer to these models.

2.3 Dependency patterns

We use the same object type corpora to derive dependency patterns. Our patterns are derived from dependency trees which are obtained using the Stanford parser[1]. Each article in each object type corpus was pre-processed by sentence splitting and named entity tagging[2]. Then each sentence was parsed by the Stanford dependency parser to obtain relational patterns. As with the chain model introduced by Sudo et al. (2001) our relational patterns are concentrated on the verbs in the sentences and contain *n+1* words (the verb and *n* words in direct or indirect relation with the verb). The number *n* is experimentally set to two words.

For illustration consider the sentence shown in Table 3 that is taken from an article in the *bridge* corpus. The first two rows of the table show the original sentence and its form after named entity tagging. The next step in processing is to replace any occurrence of a string denoting the object type by the term "OBJECTTYPE" as shown in the third row of Table 3. The final two rows of the table show the output of the Stanford dependency parser and the relational patterns identified for this example. To obtain the relational patterns from the parser output we first identified the verbs in the output. For each such verb we extracted two further words being in direct or indirect relation to the current verb. Two words are directly related if they occur in the same relational term. The verb *built-4*, for instance, is directly related to *DATE-6* because they both are in the same relational term *prep-in(built-4, DATE-6)*. Two words are indirectly related if they occur in two different terms but are linked by a word that occurs in those two terms. The verb *was-3* is, for instance, indirectly related to *OBJECTTYPE-2* because they are both in different terms but linked with *built-4* that occurs in both terms. E.g. for the term *nsubjpass(built-4, OBJECTTYPE-2)* we use the verb *built* and extract patterns based on this. *OBJECTTYPE* is in direct relation to *built* and *The* is in indirect relation to *built* through *OBJECTTYPE*. So a pattern from these relations is *The OBJECTTYPE built*. The next pattern extracted from this term is *OBJECTTYPE was built*. This pattern is based on direct relations. The verb *built* is in direct relation to *OBJECTTYPE* and also to *was*. We continue this until we cover all direct relations with *built* resulting in two more patterns (*OBJECTTYPE built DATE* and *OBJECTTYPE built W*). It should be noted that we consider all direct and indirect relations while generating the patterns.

Table 3: Example sentence for dependency pattern.

Original sentence: The bridge was built in 1876 by W. W.
After NE tagging: The bridge was built in DATE by W. W.
Input to the parser: The OBJECTTYPE was built in DATE by W. W.
Output of the parser: *det(OBJECTTYPE-2, The-1), nsubjpass(built-4, OBJECTTYPE-2), auxpass(built-4, was-3), prep-in(built-4, DATE-6), nn(W-10, W-8), agent(built-4, W-10)*
Patterns: The OBJECTTYPE built, OBJECTTYPE was built, OBJECTTYPE built DATE, OBJECTTYPE built W, was built DATE, was built W

Following these steps we extracted relational patterns for each object type corpus along with the frequency of occurrence of the pattern in the entire corpus. The frequency values are used by the summarizer to score the sentences. In the following sections we will use the term *DpM* to refer to these dependency pattern models.

2.3.1 Pattern categorization

In addition to using dependency patterns as models for biasing sentence selection, we can also use them to control the kind of information to be included in the final summary (see Section 3.2). We may want to ensure that the summary contains a sentence describing the object type of the object, its location and some background information. For example, for the object *Eiffel Tower* we aim to say that it is a tower, located in Paris, designed by Gustave Eiffel, etc. To be able to do so, we categorize dependency patterns according to the type of information they express.

We manually analyzed human written descriptions about instances of different object types and recorded for each sentence in the descriptions the kind of information it contained about the object. We analyzed descriptions of 310 different objects where each object had up to four different human written descriptions (Section 4.1). We categorized the information contained in the descriptions into

[1] http://nlp.stanford.edu/software/lex-parser.shtml

[2] For performing shallow text analysis the OpenNLP tools (http://opennlp.sourceforge.net/) were used.

the following categories:

- **type**: sentences containing the "type" information of the object such as *XXX* is a *bridge*
- **year**: sentences containing information about when the object was built or in case of mountains, for instance, when it was first climbed
- **location**: sentences containing information about where the object is located
- **background**: sentences containing some specific information about the object
- **surrounding**: sentences containing information about what other objects are close to the main object
- **visiting**: sentences containing information about e.g. visiting times, etc.

We also manually assigned each dependency pattern in each corpus-derived model to one of the above categories, provided it occurred five or more times in the object type corpora. The patterns extracted for our example sentence shown in Table 3, for instance, are all categorized by *year* category because all of them contain information about the foundation date of an object.

3 Summarizer

We adopted the same overall approach to summarization used by Aker and Gaizauskas (2009) to generate the image descriptions. The summarizer is an extractive, query-based multi-document summarization system. It is given two inputs: a toponym associated with an image and a set of documents to be summarized which have been retrieved from the web using the toponym as a query. The summarizer creates image descriptions in a three step process. First, it applies shallow text analysis, including sentence detection, tokenization, lemmatization and POS-tagging to the given input documents. Then it extracts features from the document sentences. Finally, it combines the features using a linear weighting scheme to compute the final score for each sentence and to create the final summary. We modified the approach to feature extraction and the way the summarizer acquires the weights for feature combination. The following subsections describe how feature extraction/combination is done in more detail.

3.1 Feature Extraction

The original summarizer reported in Aker and Gaizauskas (2009) uses the following features to score the sentences:

- ***querySimilarity***: Sentence similarity to the query (toponym) (cosine similarity over the vector representation of the sentence and the query).
- ***centroidSimilarity***: Sentence similarity to the centroid. The centroid is composed of the 100 most frequently occurring non stop words in the document collection (cosine similarity over the vector representation of the sentence and the centroid).
- ***sentencePosition***: Position of the sentence within its document. The first sentence in the document gets the score 1 and the last one gets $\frac{1}{n}$ where *n* is the number of sentences in the document.
- ***starterSimilarity***: A sentence gets a binary score if it starts with the query term (e.g. *Westminster Abbey*, *The Westminster Abbey*, *The Westminster* or *The Abbey*) or with the object type, e.g. *The church*. We also allow gaps (up to four words) between *the* and the query to capture cases such as *The most magnificent Abbey*, etc.
- ***LMSim***[3]: The similarity of a sentence *S* to an n-gram language model *LM* (the probability that the sentence *S* is generated by *LM*).

In our experiments we extend this feature set by two dependency pattern related features: *DpMSim* and *DepCat*.

DpMSim is computed in a similar fashion to *LMSim* feature. We assign each sentence a dependency similarity score. To compute this score, we first parse the sentence on the fly with the Stanford parser and obtain the dependency patterns for the sentence. We then associate each dependency pattern of the sentence with the occurrence frequency of that pattern in the dependency pattern model (*DpM*). *DpMSim* is then computed as given in Equation 1. It is a sum of all occurrence frequencies of the dependency patterns detected in a sentence *S* that are also contained in the *DpM*.

$$DpMSim(S, DpM) = \sum_{p \in S} f_{DpM}(p) \quad (1)$$

The second feature, *DepCat*, uses dependency patterns to categorize the sentences rather than ranking them. It can be used independently from other features to categorize each sentence by one of the categories described in Section 2.3.1. To do this, we obtain the relational patterns for the current sentence, check whether for each such pattern whether it is included in the *DpM*, and, if so, we add to the sentence the category the pattern was manually associated with. It should be noted that a sentence can have more than one category. This can occur, for instance, if the sentence contains information about when *something* was built and at the same time where it is located. It is also important to mention that assigning sentences categories does not change the order in the ranked list.

We use *DepCat* to generate an automated summary by first including sentences containing the category "type", then "year" and so on until the

[3] In Aker and Gaizauskas (2009) this feature is called *modelSimilarity*.

summary length is violated. The sentences are selected according to the order in which they occur in the ranked list. From each of the first three categories ("type", "year" and "location") we take a single sentence to avoid redundancy. The same is applied to the final two categories ("surrounding" and "visiting"). Then, if length limit is not violated, we fill the summary with sentences from the "background" category until the word limit of 200 words is reached. Here the number of added sentences is not limited. Finally, we order the sentences by first adding the sentences from the first three categories to the summary, then the "background" related sentences and finally the last two sentences from the "surrounding" and "visiting" categories. However, in cases where we have not reached the summary word limit because of uncovered categories, i.e. there were not, for instance, sentences about "location", we add to the end of the summary the next top sentence from the ranked list that was not taken.

3.2 Sentence Selection

To compute the final score for each sentence Aker and Gaizauskas (2009) use a linear function with weighted features:

$$S_{score} = (\sum_{i=1}^{n} feature_i * weight_i) \qquad (2)$$

We use the same approach, but whereas the feature weights they use are experimentally set rather than learned, we learn the weights using linear regression instead. We used $\frac{2}{3}$ of the 310 images from our image set (see Section 4.1) to train the weights. The image descriptions from this data set are used as model summaries.

Our training data contains for each image a set of image descriptions taken from the *VirtualTourist* travel community web-site [4]. From this web-site we took all existing image descriptions about a particular image or object. Note that some of these descriptions about a particular object were used to derive the model summaries for that object (see Section 4.1). Assuming that model summaries contain the most relevant sentences about an object we perform ROUGE comparisons between the sentences in all the image descriptions and the model summaries, i.e. we pair each sentence from all image descriptions about a particular place with every sentence from all the model summaries for that particular object. Sentences which are exactly the same or have common parts will score higher in ROUGE than sentences which do not have anything in common. In this way, we have for each sentence from all existing image descriptions about an object a ROUGE score[5] indicating its relevance. We also ran the summarizer for each of these sentences to compute the values for the different features. This gives information about each feature's value for each sentence. Then the ROUGE scores and feature score values for every sentence were input to the linear regression algorithm to train the weights.

Given the weights, Equation 2 is used to compute the final score for each sentence. The final sentence scores are used to sort the sentences in the descending order. This sorted list is then used by the summarizer to generate the final summary as described in Aker and Gaizauskas (2009).

4 Evaluation

To evaluate our approach we used two different assessment methods: ROUGE (Lin, 2004) and manual readability. In the following we first describe the data sets used in each of these evaluations, and then we present the results of each assessment.

4.1 Data sets

For evaluation we use the image collection described in Aker and Gaizauskas (2010). The image collection contains 310 different images with manually assigned toponyms. The images cover 60 of the 107 object types identified from Wikipedia (see Table 2). For each image there are up to four short descriptions or model summaries. The model summaries were created manually based on image descriptions taken from *VirtualTourist* and contain a minimum of 190 and a maximum of 210 words. An example model summary about the *Eiffel Tower* is shown in Table 4. $\frac{2}{3}$ of this image collection was used to train the weights and the remaining $\frac{1}{3}$ (105 images) for evaluation.

To generate automatic captions for the images we automatically retrieved the top 30 related web-documents for each image using the Yahoo! search engine and the toponym associated with the image as a query. The text from these documents was extracted using an HTML parser and passed to the summarizer. The set of documents we used to generate our summaries excluded any *VirtualTourist* related sites, as these were used to generate

[4] www.virtualtourist.com

[5] We used ROUGE 1.

Table 4: Model, Wikipedia baseline and starterSimilarity+LMSim+DepCat summary for Eiffel Tower.

Model Summary	Wikipedia baseline summary	starterSimilarity+LMSim+DepCat summary
The Eiffel Tower is the most famous place in Paris. It is made of 15,000 pieces fitted together by 2,500,000 rivets. It's of 324 m (1070 ft) high structure and weighs about 7,000 tones. This world famous landmark was built in 1889 and was named after its designer, engineer Gustave Alexandre Eiffel. It is now one of the world's biggest tourist places which is visited by around 6,5 million people yearly. There are three levels to visit: Stages 1 and 2 which can be reached by either taking the steps (680 stairs) or the lift, which also has a restaurant "Altitude 95" and a Souvenir shop on the first floor. The second floor also has a restaurant "Jules Verne". Stage 3, which is at the top of the tower can only be reached by using the lift. But there were times in the history when Tour Eiffel was not at all popular, when the Parisians thought it looked ugly and wanted to pull it down. The Eiffel Tower can be reached by using the Mtro through Trocadro, Ecole Militaire, or Bir-Hakeim stops. The address is: Champ de Mars-Tour Eiffel.	The Eiffel Tower (French: Tour Eiffel, [tur efel]) is a 19th century iron lattice tower located on the Champ de Mars in Paris that has become both a global icon of France and one of the most recognizable structures in the world. The Eiffel Tower, which is the tallest building in Paris, is the single most visited paid monument in the world; millions of people ascend it every year. Named after its designer, engineer Gustave Eiffel, the tower was built as the entrance arch for the 1889 World's Fair. The tower stands at 324 m (1,063 ft) tall, about the same height as an 81-story building. It was the tallest structure in the world from its completion until 1930, when it was eclipsed by the Chrysler Building in New York City. Not including broadcast antennas, it is the second-tallest structure in France, behind the Millau Viaduct, completed in 2004. The tower has three levels for visitors. Tickets can be purchased to ascend either on stairs or lifts to the first and second levels.	The Eiffel Tower, which is the tallest building in Paris, is the single most visited paid monument in the world; millions of people ascend it every year. The tower is located on the Left Bank of the Seine River, at the northwestern extreme of the Parc du Champ de Mars, a park in front of the Ecole Militaire that used to be a military parade ground. The tower was met with much criticism from the public when it was built, with many calling it an eyesore. Counting from the ground, there are 347 steps to the first level, 674 steps to the second level, and 1,710 steps to the small platform on the top of the tower. Although it was the world's tallest structure when completed in 1889, the Eiffel Tower has since lost its standing both as the tallest lattice tower and as the tallest structure in France. The tower has two restaurants: Altitude 95, on the first floor 311ft (95m) above sea level; and the Jules Verne, an expensive gastronomical restaurant on the second floor, with a private lift.

Table 5: ROUGE scores for each single feature and Wikipedia baseline.

Recall	centroidSimilarity	sentencePosition	querySimilarity	starterSimilarity	LMSim	DpMSim***	Wiki
R2	.0734	**.066**	.0774	.0869	.0895	**.093**	.097
RSU4	.12	**.11**	.12	.137	.142	**.145**	.14

the model summaries.

4.2 ROUGE assessment

In the first assessment we compared the automatically generated summaries against model summaries written by humans using ROUGE (Lin, 2004). Following the Document Understanding Conference (DUC) evaluation standards we used ROUGE 2 (*R2*) and ROUGE SU4 (*RSU4*) as evaluation metrics (Dang, 2006) . ROUGE 2 gives recall scores for bi-gram overlap between the automatically generated summaries and the reference ones. ROUGE SU4 allows bi-grams to be composed of non-contiguous words, with a maximum of four words between the bi-grams.

As baselines for evaluation we used two different summary types. Firstly, we generated summaries for each image using the top-ranked non Wikipedia document retrieved in the Yahoo! search results for the given toponyms. From this document we create a baseline summary by selecting sentences from the beginning until the summary reaches a length of 200 words. As a second baseline we use the Wikipedia article for a given toponym from which we again select sentences from the beginning until the summary length limit is reached.

First, we compared the baseline summaries against the VirtualTourist model summaries. The comparison shows that the Wikipedia baseline ROUGE scores (R2 .097***, RSU4 .14***) are significantly higher than the first document ones (R2 0.042, RSU4 .079) [6]. Thus, we will focus on the Wikipedia baseline summaries to draw conclusions about our automatic summaries. Table 4 shows the Wikipedia baseline summary about the *Eiffel Tower*.

Secondly, we separately ran the summarizer over the top ten documents for each single feature and compared the automated summaries against the model ones. The results of this comparison are shown in Table 5.

Table 5 shows that the dependency model feature (*DpMSim*) contributes most to the summary quality according to the ROUGE metrics. It is also significantly better than all other feature scores except the *LMSim* feature. Compared to *LMSim* ROUGE scores the *DpMSim* feature offers only a moderate improvement. The same moderate improvement we can see between the *DpMSim* RSU4 and the *Wiki* RSU4. The lowest ROUGE scores are obtained if only sentence position (*sentecePosition*) is used.

To see how the ROUGE scores change when features are combined with each other we performed different combinations of the features, ran the summarizer for each combination and compared the automated summaries against the model ones. In the different combinations we

[6] To assess the statistical significance of ROUGE score differences between multiple summarization results we performed a pairwise Wilcoxon signed-rank test. We use the following conventions for indicating significance level in the tables: *** = $p < .0001$, ** = $p < .001$, * = $p < .05$ and no star indicates non-significance.

Table 6: ROUGE scores of feature combinations which score moderately or significantly higher than dependency pattern model (*DpMSim*) feature and Wikipedia baseline.

Recall	starterSimilarity + LMSim	starterSimilarity + LMSim + DepCat***	DpmSim	Wiki
R2	.095	**.102**	.093	.097
RSU4	.145	**.155**	.145	.14

also included the dependency pattern categorization (*DepCat*) feature explained in Section 3.1. Table 6 shows the results of feature combinations which score moderately or significantly higher than the dependency pattern model (*DpMSim*) feature score shown in Table 5.

The results showed that combining *DpMSim* with other features did not lead to higher ROUGE scores than those produced by that feature alone.

The summaries categorized by dependency patterns (*starterSimilarity+LMSim+DepCat*) achieve significantly higher ROUGE scores than the Wikipedia baseline. For both ROUGE R2 and ROUGE SU4 the significance is at level $p < .0001$. Table 4 shows a summary about the *Eiffel Tower* obtained using this *starterSimilarity+LMSim+DepCat* feature. Table 5 also shows the ROUGE scores of the feature combination *starterSimilarity* and *LMSim* used without the dependency categorization (*DepCat*) feature. It can be seen that this combination without the dependency patterns lead to lower ROUGE scores in ROUGE 2 and only moderate improvement in ROUGE SU4 if compared with Wikipedia baseline ROUGE scores.

4.3 Readability assessment

We also evaluated our summaries using a readability assessment as in DUC and TAC. DUC and TAC manually assess the quality of automatically generated summaries by asking human subjects to score each summary using five criteria – *grammaticality, redundancy, clarity, focus* and *structure* criteria. Each criterion is scored on a five point scale with high scores indicating a better result (Dang, 2005).

For this evaluation we used the same 105 images as in the ROUGE evaluation. As the ROUGE evaluation showed that the dependency pattern categorization (*DepCat*) renders the best results when used in feature combination *starterSimilarity + LMSim + DepCat*, we further investigated the contribution of dependency pattern categorization by performing a readability assessment on summaries generated using this feature combination. For comparison we also evaluated summaries which were not structured by dependency patterns (*starterSimilarity + LMSim*) and also the Wikipedia baseline summaries.

We asked four people to assess the summaries. Each person was shown all 315 summaries (105 from each summary type) in a random way and was asked to assess them according to the DUC and TAC manual assessment scheme. The results are shown in Table 7.

We see from Table 7 that using dependency patterns to categorize the sentences and produce a structured summary helps to obtain better readable summaries. Looking at the 5 and 4 scores the table shows that the dependency pattern categorized summaries (*SLMD*) have better clarity (85% of the summaries), are more coherent (74% of the summaries), contain less redundant information (83% of the summaries) and have better grammar (92% of the summaries) than the ones without dependency categorization (80%, 70%, 60%, 84%).

The scores of our automated summaries were better than the Wikipedia baseline summaries in the *grammar* feature. However, in other features the Wikipedia baseline summaries obtained better scores than our automated summaries. This comparison show that there is a gap to fill in order to obtain better readable summaries.

5 Related Work

Our approach has an advantage over related work in automatic image captioning in that it requires only GPS information associated with the image in order to generate captions. Other attempts towards automatic generation of image captions generate captions based on the immediate textual context of the image with or without consideration of image related features such as colour, shape or texture (Deschacht and Moens, 2007; Mori et al., 2000; Barnard and Forsyth, 2001; Duygulu et al., 2002; Barnard et al., 2003; Pan et al., 2004; Feng and Lapata, 2008; Satoh et al., 1999; Berg et al., 2005). However, Marsch & White (2003) argue that the content of an image and its immediate text have little semantic agreement and this can, according to Purves et al. (2008), be misleading to image retrieval. Furthermore, these approaches assume that the image has been obtained from a document. In cases where there is no document associated with the image, which is the scenario we are principally concerned with, these techniques are not applicable.

Table 7: Readability evaluation results: Each cell shows the percentage of summaries scoring the ranking score heading the column for each criterion in the row as produced by the summary method indicated by the subcolumn heading – Wikipedia baseline (W), starterSimilarity + LMSim (SLM) and starterSimilarity + LMSim + DepCat (SLMD). The numbers indicate the percentage values averaged over the four people.

	5			4			3			2			1		
Criterion	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD	W	SLM	SLMD
clarity	72.6	50.5	53.6	21.7	30.0	31.4	1.2	6.7	5.7	4.0	10.2	6.0	0.5	2.6	3.3
focus	72.1	49.3	51.2	20.5	26.0	25.2	3.8	10.0	10.7	3.3	10.0	10.5	0.2	4.8	2.4
coherence	67.1	39.0	48.3	23.6	31.4	26.9	4.8	12.4	11.9	3.3	10.2	9.8	1.2	6.9	3.1
redundancy	69.8	42.9	55.0	21.7	17.4	28.8	2.4	4.5	4.3	5.0	27.1	8.8	1.2	8.1	3.1
grammar	48.6	55.7	62.9	32.9	29.0	30.0	5.0	3.1	1.9	11.7	12.1	5.2	1.9	0	0

Dependency patterns have been exploited in various language processing applications. In information extraction, for instance, dependency patterns have been used to extract relevant information from text resources (Yangarber et al., 2000; Sudo et al., 2001; Culotta and Sorensen, 2004; Stevenson and Greenwood, 2005; Bunescu and Mooney, 2005; Stevenson and Greenwood, 2009). However, dependency patterns have not been used extensively in summarization tasks. We are aware only of the work described in Nobata et al. (2002) who used dependency patterns in combination with other features to generate extracts in a single document summarization task. The authors found that when learning weights in a simple feature weigthing scheme, the weight assigned to dependency patterns was lower than that assigned to other features. The small contribution of the dependency patterns may have been due to the small number of documents they used to derive their dependency patterns – they gathered dependency patterns from only ten domain specific documents which are unlikely to be sufficient to capture repeated features in a domain.

6 Discussion and Conclusion

We have proposed a method by which dependency patterns extracted from corpora of descriptions of instances of particular object types can be used in a multi-document summarizer to automatically generate image descriptions. Our evaluations show that such an approach yields summaries which score more highly than an approach which uses a simpler representation of an object type model in the form of a n-gram language model.

When used as the sole feature for sentence ranking, dependency pattern models (*DpMSim*) produced summaries with higher ROUGE scores than those obtained using the features reported in Aker and Gaizauskas (2009). These dependency pattern models also achieved a modest improvement over Wikipedia baseline ROUGE SU4. Furthermore, we showed that using dependency patterns in combination with features reported in Aker and Gaizauskas to produce a structured summary led to significantly better results than Wikipedia baseline summaries as assessed by ROUGE. However, human assessed readability showed that there is still scope for improvement.

These results indicate that dependency patterns are worth investigating for object focused automated summarization tasks. Such investigations should in particular concentrate on how dependency patterns can be used to structure information within the summary, as our best results were achieved when dependency patterns were used for this purpose.

There are a number of avenues to pursue in future work. One is to explore how dependency patterns could be used to produce generative summaries and/or perform sentence trimming. Another is to investigate how dependency patterns might be automatically clustered into groups expressing similar or related facts, rather than relying on manual categorization of dependency patterns into categories such as “type”, “year”, etc. as was done here. Evaluation should be extended to investigate the utility of the automatically generated image descriptions for image retrieval. Finally, we also plan to analyze automated ways for learning information structures (e.g. what is the flow of facts to describe a location) from existing image descriptions to produce better summaries.

7 Acknowlegment

The research reported was funded by the TRIPOD project supported by the European Commission under the contract No. 045335. We would like to thank Emina Kurtic, Mesude Bicak, Edina Kurtic and Olga Nesic for participating in our manual evaluation. We also would like to thank Trevor Cohn and Mark Hepple for discussions and comments.

References

A. Aker and R. Gaizauskas. 2009. Summary Generation for Toponym-Referenced Images using Object

Type Language Models. *International Conference on Recent Advances in Natural Language Processing (RANLP),2009*.

A. Aker and R. Gaizauskas. 2010. Model Summaries for Location-related Images. In *Proc. of the LREC-2010 Conference*.

K. Barnard and D. Forsyth. 2001. Learning the semantics of words and pictures. In *International Conference on Computer Vision*, volume 2, pages 408–415. Vancouver: IEEE.

K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D.M. Blei, and M.I. Jordan. 2003. Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135.

T.L. Berg, A.C. Berg, J. Edwards, and DA Forsyth. 2005. Whos in the Picture? In *Advances in Neural Information Processing Systems 17: Proc. Of The 2004 Conference*. MIT Press.

R.C. Bunescu and R.J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics Morristown, NJ, USA.

A. Culotta and J. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 423–429, Barcelona, Spain, July.

H.T. Dang. 2005. Overview of DUC 2005. *DUC 05 Workshop at HLT/EMNLP*.

H.T. Dang. 2006. Overview of DUC 2006. *National Institute of Standards and Technology*.

K. Deschacht and M.F. Moens. 2007. Text Analysis for Automatic Image Annotation. *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics. East Stroudsburg: ACL.*

P. Duygulu, K. Barnard, JFG de Freitas, and D.A. Forsyth. 2002. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. *In Seventh European Conference on Computer Vision (ECCV)*, 4:97–112.

X. Fan, A. Aker, M. Tomko, P. Smart, M Sanderson, and R. Gaizauskas. 2010. Automatic Image Captioning From the Web For GPS Photographs. In *Proc. of the 11th ACM SIGMM International Conference on Multimedia Information Retrieval, National Constitution Center, Philadelphia, Pennsylvania*.

Y. Feng and M. Lapata. 2008. Automatic Image Annotation Using Auxiliary Text Information. *Proc. of Association for Computational Linguistics (ACL) 2008, Columbus, Ohio, USA*.

C.Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. *Proc. of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

E.E. Marsh and M.D. White. 2003. A taxonomy of relationships between images and text. *Journal of Documentation*, 59:647–672.

Y. Mori, H. Takahashi, and R. Oka. 2000. Automatic word assignment to images based on image division and vector quantization. In *Proc. of RIAO 2000: Content-Based Multimedia Information Access*.

C. Nobata, S. Sekine, H. Isahara, and R. Grishman. 2002. Summarization system integrated with named entity tagging and ie pattern discovery. In *Proc. of the LREC-2002 Conference*, pages 1742–1745.

J.Y. Pan, H.J. Yang, P. Duygulu, and C. Faloutsos. 2004. Automatic image captioning. In *Multimedia and Expo, 2004. ICME'04. IEEE International Conference on*, volume 3.

RS Purves, A. Edwardes, and M. Sanderson. 2008. Describing the where–improving image annotation and search through geography. *1st Intl. Workshop on Metadata Mining for Image Understanding, Funchal, Madeira-Portugal*.

S. Satoh, Y. Nakamura, and T. Kanade. 1999. Name-It: naming and detecting faces in news videos. *Multimedia, IEEE*, 6(1):22–35.

F. Song and W.B. Croft. 1999. A general language model for information retrieval. In *Proc. of the eighth international conference on Information and knowledge management*, pages 316–321. ACM New York, NY, USA.

M. Stevenson and M.A. Greenwood. 2005. A semantic approach to IE pattern induction. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386. Association for Computational Linguistics Morristown, NJ, USA.

M. Stevenson and M. Greenwood. 2009. Dependency Pattern Models for Information Extraction. *Research on Language and Computation*, 7(1):13–39.

K. Sudo, S. Sekine, and R. Grishman. 2001. Automatic pattern acquisition for Japanese information extraction. In *Proc. of the first international conference on Human language technology research*, page 7. Association for Computational Linguistics.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proc. of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 940–946. Saarbriicken, Germany, August.

Incorporating Extra-linguistic Information into Reference Resolution in Collaborative Task Dialogue

Ryu Iida **Shumpei Kobayashi** **Takenobu Tokunaga**
Tokyo Institute of Technology
2-12-1, Ôokayama, Meguro, Tokyo 152-8552, Japan
{ryu-i,skobayashi,take}@cl.cs.titech.ac.jp

Abstract

This paper proposes an approach to reference resolution in situated dialogues by exploiting extra-linguistic information. Recently, investigations of referential behaviours involved in situations in the real world have received increasing attention by researchers (Di Eugenio et al., 2000; Byron, 2005; van Deemter, 2007; Spanger et al., 2009). In order to create an accurate reference resolution model, we need to handle extra-linguistic information as well as textual information examined by existing approaches (Soon et al., 2001; Ng and Cardie, 2002, etc.). In this paper, we incorporate extra-linguistic information into an existing corpus-based reference resolution model, and investigate its effects on reference resolution problems within a corpus of Japanese dialogues. The results demonstrate that our proposed model achieves an accuracy of 79.0% for this task.

1 Introduction

The task of identifying reference relations including anaphora and coreferences within texts has received a great deal of attention in natural language processing, from both theoretical and empirical perspectives. Recently, research trends for reference resolution have drastically shifted from handcrafted rule-based approaches to corpus-based approaches, due predominately to the growing success of machine learning algorithms (such as Support Vector Machines (Vapnik, 1998)); many researchers have examined ways for introducing various linguistic clues into machine learning-based models (Ge et al., 1998; Soon et al., 2001; Ng and Cardie, 2002; Yang et al., 2003; Iida et al., 2005; Yang et al., 2005; Yang et al., 2008; Poon and Domingos, 2008, etc.). Research has continued to progress each year, focusing on tackling the problem as it is represented in the annotated data sets provided by the Message Understanding Conference (MUC)[1] and the Automatic Content Extraction (ACE)[2]. In these data sets, coreference relations are defined as a limited version of a typical coreference; this generally means that only the relations where expressions refer to the same named entities are addressed, because it makes the coreference resolution task more information extraction-oriented. In other words, the coreference task as defined by MUC and ACE is geared toward only identifying coreference relations anchored to an entity within the text.

In contrast to this research trend, investigations of referential behaviour in real world situations have continued to gain interest in the language generation community (Di Eugenio et al., 2000; Byron, 2005; van Deemter, 2007; Foster et al., 2008; Spanger et al., 2009), aiming at applications such as human-robot interaction. Spanger et al. (2009) for example constructed a corpus by recording dialogues of two participants collaboratively solving the Tangram puzzle. The corpus includes extra-lingustic information synchronised with utterances (such as operations on the puzzle pieces). They analysed the relations between referring expressions and the extra-linguistic information, and reported that the pronominal usage of referring expressions is predominant. They also revealed that the multi-modal perspective of reference should be dealt with for more realistic reference understanding. Thus, a challenging issue in reference resolution is to create a model bridging a referring expression in the text and its object in the real world. As a first step, this paper focuses on incorporating extra-linguistic information into an existing corpus-based approach, taking Spanger et al. (2009)'s REX-J corpus[3] as the data set. In our

[1] www-nlpir.nist.gov/related_projects/muc/
[2] www.itl.nist.gov/iad/mig//tests/ace/
[3] The corpus was named REX-J after their publication of

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1259–1267,
Uppsala, Sweden, 11-16 July 2010.

problem setting, a referent needs to be identified by taking into account extra-linguistic information, such as the spatiala relations of puzzle pieces and the participants' operations on them, as well as any preceding utterances in the dialogue. We particularly focus on the participants' operation of pieces and so introduce it as several features in a machine learning-based approach.

This paper is organised as follows. We first explain the corpus of collaborative work dialogues in Section 2, and then present our approach for identifying a referent given a referring expression in situated dialogues in Section 3. Section 4 shows the results of our empirical evaluation. In Section 5 we compare our work with existing work on reference resolution, and then conclude this paper and discuss future directions in Section 6.

2 REX-J corpus: a corpus of collaborative work dialogue

For investigating dialogue from the multi-modal perspective, researchers have developed data sets including extra-linguistic information, bridging objects in the world and their referring expressions. The COCONUT corpus (Di Eugenio et al., 2000) is collected from keyboard-dialogues between two participants, who are collaborating on a simple 2D design task. The setting tends to encourage simple types of expressions by the participants. The COCONUT corpus is also limited to annotations with symbolic information about objects, such as object attributes and location in discrete coordinates. Thus, in addition to the artificial nature of interaction, such as using keyboard input, this corpus only records restricted types of data.

On the other hand, though the annotated corpus by Spanger et al. (2009) focuses on a limited domain (i.e. collaborative work dialogues for solving the Tangram puzzle using a puzzle simulator on the computer), the required operations to solve the puzzle, and the situation as it is updated by a series of operations on the pieces are both recorded by the simulator. The relationship between a referring expression in a dialogue and its referent on a computer display is also annotated. For this reason, we selected the REX-J corpus for use in our empirical evaluations on reference resolution. Before explaining the details of our evaluation, we sketch out the REX-J corpus and some of its prominent statistics.

Spanger et al. (2009), which describes its construction.

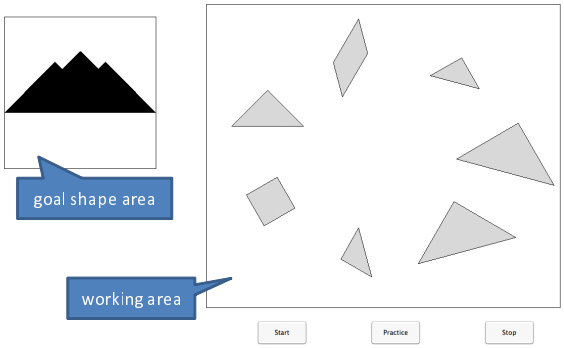


Figure 1: Screenshot of the Tangram simulator

2.1 The REX-J corpus

In the process of building the REX-J corpus, Spanger et al. (2009) recruited 12 Japanese graduate students (4 females and 8 males), and split them into 6 pairs. All pairs knew each other previously and were of the same sex and approximately the same age. Each pair was instructed to solve the Tangram puzzle. The goal of the puzzle is to construct a given shape by arranging seven pieces of simple figures as shown in Figure 1. The precise position of every piece and every action that the participants make are recorded by the Tangram simulator in which the pieces on the computer display can be moved, rotated and flipped with simple mouse operations. The piece position and the mouse actions were recorded at intervals of 10 msec. The simulator displays two areas: a goal shape area (the left side of Figure 1) and a working area (the right side of Figure 1) where pieces are shown and can be manipulated.

A different role was assigned to each participant of a pair: a *solver* and an *operator*. Given a certain goal shape, the solver thinks of the necessary arrangement of the pieces and gives instructions to the operator for how to move them. The operator manipulates the pieces with the mouse according to the solver's instructions. During this interaction, frequent uttering of referring expressions are needed to distinguish the pieces of the puzzle. This collaboration is achieved by placing a set of participants side by side, each with their own display showing the work area, and a shield screen set between them to prevent the operator from seeing the goal shape, which is visible only on the solver's screen, and to further restrict their

interaction to only speech.

2.2 Statistics

Table 1 lists the syntactic and semantic features of the referring expressions in the corpus with their respective frequencies. Note that multiple features can be used in a single expression. This list demonstrates that 'pronoun' and 'shape' features are frequently uttered in the corpus. This is because pronominal expressions are often used for pointing to a piece on a computer display. Expressions representing 'shape' frequently appear in dialogues even though they may be relatively redundant in the current utterance. From these statistics, capturing these two features can be judged as crucial as a first step toward accurate reference resolution.

3 Reference Resolution using Extra-linguistic Information

Before explaining the treatment of extra-linguistic information, let us first describe the task definition, taking the REX-J corpus as target data. In the task of reference resolution, the reference resolution model has to identify a referent (i.e. a piece on a computer display)[4]. In comparison to conventional problem settings for anaphora resolution, where the model searches for an antecedent out of a set of candidate antecedents from preceding utterances, expressions corresponding to antecedents are sometimes omitted because referring expressions are used as deixis (i.e. physically pointing to a piece on a computer display); they may also refer to a piece that has just been manipulated by an operator due to the temporal salience in a series of operations. For these reasons, even though the model checks all candidates in the preceding utterances, it may not find the antecedent of a given referring expression. However, we do know that each referent exists as a piece on the display. We can therefore establish that when a referring expression is uttered by either a solver or an operator, the model can choose one of seven pieces as a referent of the current referring expression.

3.1 Ranking model to identify referents

To investigate the impact of extra-linguistic information on reference resolution, we conduct an empirical evaluation in which a reference resolution model chooses a referent (i.e. a piece) for a given referring expression from the set of pieces illustrated on the computer display.

As a basis for our reference resolution model, we adopt an existing model for reference resolution. Recently, machine learning-based approaches to reference resolution (Soon et al., 2001; Ng and Cardie, 2002, etc.) have been developed, particularly focussing on identifying anaphoric relations in texts, and have achieved better performance than hand-crafted rule-based approaches. These models for reference resolution take into account linguistic factors, such as relative salience of candidate antecedents, which have been modeled in Centering Theory (Grosz et al., 1995) by ranking candidate antecedents appearing in the preceding discourse (Iida et al., 2003; Yang et al., 2003; Denis and Baldridge, 2008). In order to take advantage of existing models, we adopt the ranking-based approach as a basis for our reference resolution model.

In conventional ranking-based models, Yang et al. (2003) and Iida et al. (2003) decompose the ranking process into a set of pairwise comparisons of two candidate antecedents. However, recent work by Denis and Baldridge (2008) reports that appropriately constructing a model for ranking all candidates yields improved performance over those utilising pairwise ranking.

Similarly we adopt a *ranking-based* model, in which all candidate antecedents compete with one another to decide the most likely candidate antecedent. Although the work by Denis and Baldridge (2008) uses Maximum Entropy to create their ranking-based model, we adopt the Ranking SVM algorithm (Joachims, 2002), which learns a weight vector to rank candidates for a given partial ranking of each referent. Each training instance is created from the set of all referents for each referring expression. To define the partial ranking of referents, we simply rank referents referred to by a given referring expression as first place and other referents as second place.

3.2 Use of extra-linguistic information

Recent work on multi-modal reference resolution or referring expression generation (Prasov and Chai, 2008; Foster et al., 2008; Carletta et al., 2010) indicates that extra-linguistic information, such as eye-gaze and manipulation of objects, is

[4] In the current task on reference resolution, we deal only with referring expressions referring to a single piece to minimise complexity.

Table 1: Referring expressions in REX-J corpus

feature	tokens	example
demonstratives	742	
adjective	194	"*ano migigawa no sankakkei* (that triangle at the right side)"
pronoun	548	"*kore* (this)"
attribute	795	
size	223	"*tittyai sankakkei* (the small triangle)"
shape	566	"*ôkii sankakkei* (the large triangle)"
direction	6	"*ano sita muiteru dekai sankakkei* (that large triangle facing to the bottom)"
spatial relations	147	
projective	143	"*hidari no okkii sankakkei* (the small triangle on the left)"
topological	2	"*ôkii hanareteiru yatu* (the big distant one)"
overlapping	2	" *sono sita ni aru sankakkei* (the triangle underneath it)"
action-mentioning	85	"*migi ue ni doketa sankakkei* (the triangle you put away to the top right)"

one of essential clues for distinguishing deictic reference from endophoric reference.

For instance, Prasov and Chai (2008) demonstrated that integrating eye-gaze information (especially, relative fixation intensity, the amount of time spent fixating a candidate object) into the conventional dialogue history-based model improved the performance of reference resolution. Foster et al. (2008) investigated the relationship of referring expressions and the manupluation of objects on a collaborative construction task, which is similar to our Tangram task[5]. They reported about 36% of the initial mentioned referring expressions in their corpus were involved with participant's operations of objects, such as mouse manipulation.

From these background, in addition to the information about the history of the preceding discourse, which has been used in previous machine learning-based approaches, we integrate extra-linguistic information into the reference resolution model shown in Section 3.1. More precisely, we introduce the following extra-linguistic information: the information with regards to the history of a piece's movement and the mouse cursor positions, and the information of the piece currently manipulated by an operator. We next elaborate on these three kinds of features. All the features are summarised in Table 2.

3.2.1 Discourse history features

First, 'type of' features are acquired from the expressions of a given referring expression and its antecedent in the preceding discourse if the antecedent explicitly appears. These features have been examined by approaches to anaphora or coreference resolution (Soon et al., 2001; Ng and Cardie, 2002, etc.) to capture the salience of a candidate antecedent. To capture the textual aspect of dialogues for solving Tangram puzzle, we exploit the features such as a binary value indicating whether a referring expression has no antecedent in the preceding discourse and case markers following a candidate antecedent.

3.2.2 Action history features

The history of the operations may yield important clues that indicate the salience in terms of the temporal recency of a piece within a series of operations. To introduce this aspect as a set of features, we can use, for example, the time distance of a candidate referent (i.e. a piece in the Tangram puzzle) since the mouse cursor was moved over it. We call this type of feature the *action history feature*.

3.2.3 Current operation features

The recency of operations of a piece is also an important factor on reference resolution because it is directly associated with the focus of attention in terms of the cognition in a series of operations. For example, since a piece which was most recently manipulated is most salient from cognitive perspectives, it might be expected that the piece tends to be referred to by unmarked referring expressions such as pronouns. To incorporate such clues into the reference resolution model, we can use, for example, the time distance of a candidate referent since it was last manipulated in the preceding utterances. We call this type of feature the *current operation feature*.

[5]Note that the task defined in Foster et al. (2008) makes no distinction between two roles; a operator and a solver. Thus, two partipants both can mamipulate pieces on a computer display, but need to jointly construct to create a predefined goal shape.

Table 2: Feature set

(a) **Discourse history features**		
DH1 :	yes, no	a binary value indicating that P is referred to by the most recent referring expression.
DH2 :	yes, no	a binary value indicating that the time distance to the last mention of P is less than or equal to 10 sec.
DH3 :	yes, no	a binary value indicating that the time distance to the last mention of P is more than 10 sec and less than or equal to 20 sec.
DH4 :	yes, no	a binary value indicating that the time distance to the last mention of P is more than 20 sec.
DH5 :	yes, no	a binary value indicating that P has never been referred to by any mentions in the preceding utterances.
DH6 :	yes, no, N/A	a binary value indicating that the attributes of P are compatible with the attributes of R.
DH7 :	yes, no	a binary value indicating that R is followed by the case marker '*o* (accusative)'.
DH8 :	yes, no	a binary value indicating that R is followed by the case marker '*ni* (dative)'.
DH9 :	yes, no	a binary value indicating that R is a pronoun and the most recent reference to P is not a pronoun.
DH10 :	yes, no	a binary value indicating that R is not a pronoun and was most recently referred to by a pronoun.
(b) **Action history features**		
AH1 :	yes, no	a binary value indicating that the mouse cursor was over P at the beginning of uttering R.
AH2 :	yes, no	a binary value indicating that P is the last piece that the mouse cursor was over when feature AH1 is 'no'.
AH3 :	yes, no	a binary value indicating that the time distance is less than or equal to 10 sec after the mouse cursor was over P.
AH4 :	yes, no	a binary value indicating that the time distance is more than 10 sec and less than or equal to 20 sec after the mouse cursor was over P.
AH5 :	yes, no	a binary value indicating that the time distance is more than 20 sec after the mouse cursor was over P.
AH6 :	yes, no	a binary value indicating that the mouse cursor was never over P in the preceding utterances.
(c) **Current operation features**		
CO1 :	yes, no	a binary value indicating that P is being manipulated at the beginning of uttering R.
CO2 :	yes, no	a binary value indicating that P is the most recently manipulated piece when feature CO1 is 'no'.
CO3 :	yes, no	a binary value indicating that the time distance is less than or equal to 10 sec after P was most recently manipulated.
CO4 :	yes, no	a binary value indicating that the time distance is more than 10 sec and less than or equal to 20 sec after P was most recently manipulated.
CO5 :	yes, no	a binary value indicating that the time distance is more than 20 sec after P was most recently manipulated.
CO6 :	yes, no	a binary value indicating that P has never been manipulated.

P stands for a piece of the Tangram puzzle (i.e. a candidate referent of a referring expression) and R stands for the target referring expression.

4 Empirical Evaluation

In order to investigate the effect of the extra-linguistic information introduced in this paper, we conduct an empirical evaluation using the REX-J corpus.

4.1 Models

As we see in Section 2.2, the feature testing whether a referring expression is a pronoun or not is crucial because it is directly related to the 'deictic' usage of referring expressions, whereas other expressions tend to refer to an expression appearing in the preceding utterances. As described in Denis and Baldridge (2008), when the size of training instances is relatively small, the models induced by learning algorithms (e.g. SVM) should be separately created with regards to distinct features. Therefore, focusing on the difference of the pronominal usage of referring expressions, we separately create the reference resolution models; one is for identifying a referent of a given pronoun, and the other is for all other expressions. We henceforth call the former model the *pronoun model* and the latter one the *non-pronoun model* respectively. At the training phase, we use only training instances whose referring expressions are pronouns for creating the pronoun model, and all other training instances are used for the non-pronoun model. The model using one of these models depending on the referring expression to be solved is called the *separate model*.

To verify Denis and Baldridge (2008)'s premise mentioned above, we also create a model using all training instances without dividing pronouns and other. This model is called the *combined model* hereafter.

4.2 Experimental setting

We used 40 dialogues in the REX-J corpus[6], containing 2,048 referring expressions. To facilitate the experiments, we conduct 10-fold crossvalidation using 2,035 referring expressions, each of which refers to a single piece in a computer dis-

[6]Spanger et al. (2009)'s original corpus contains only 24 dialogues. In addition to this, we obtained anothor 16 dialogues by favour of the authors.

Table 3: Results on reference resolution: accuracy

model	discourse history (baseline)		+action history*		+current operation		+action history, +current operation*	
separated model (a+b)	0.664	(1352/2035)	0.790	(1608/2035)	0.685	(1394/2035)	0.780	(1587/2035)
a) pronoun model	0.648	(660/1018)	0.886	(902/1018)	0.692	(704/1018)	0.875	(891/1018)
b) non-pronoun model	0.680	(692/1017)	0.694	(706/1017)	0.678	(690/1017)	0.684	(696/1017)
combined model	0.664	(1352/2035)	0.749	(1524/2035)	0.650	(1322/2035)	0.743	(1513/2035)

'*' means the extra-lingustic features (or the combinations of them) significantly contribute to improving performance. For the significant tests, we used McNemar test with Bonferroni's correction for multiple comparisons, i.e. $\alpha/K = 0.05/4 = 0.01$.

play[7].

As a baseline model, we adopted a model only using the discourse history features. We utilised *SVM*^{rank}[8] as an implementation of the Ranking SVM algorithm, in which the parameter *c* was set as 1.0 and the remaining parameters were set to their defaults.

4.3 Results

The results of each model are shown in Table 3. First of all, by comparing the models with and without extra-linguistic information (i.e. the model using all features shown in Table 2 and the baseline model), we can see the effectiveness of extra-linguistic information. The results typically show that the former achieved better performance than the latter. In particular, it indicates that exploiting the action history features are significantly useful for reference resolution in this data set.

Second, we can also see the impact of extra-linguistic information (especially, the action history features) with regards to the pronoun and non-pronoun models. In the former case, the model with extra-linguistic information improved by about 22% compared with the baseline model. On the other hand, in the latter case, the accuracy improved by only 7% over the baseline model. The difference may be caused by the fact that pronouns are more sensitive to the usage of the action history features because pronouns are often uttered as deixis (i.e. a pronoun tends to directly refer to a piece shown in a computer display).

The results also show that the model using the discourse history and action history features achieved better performance than the model using all the features. This may be due to the duplicated definitions between the action history and current operation features. As we can see in the feature definitions of CO1 and AH1, some current operation features partially overlap with the action history features, which is effectively used in the ranking process. However, the other current operation features may have bad effects for ranking referents due to their ill-formed definitions. To shed light on this problem, we need additional investigation of the usage of features, and to refine their definitions.

Finally, the results show that the performance of the separated model is significantly better than that of the combined model[9], which indicates that separately creating models to specialise in distinct factors (i.e. whether a referring expression is a pronoun or not) is important as suggested by Denis and Baldridge (2008).

We next investigated the significance of each

Table 4: Weights of the features in each model

	pronoun model		non-pronoun model	
rank	feature	weight	feature	weight
1	AH1	0.6371	DH6	0.7060
2	AH3	0.2721	DH2	0.2271
3	DH1	0.2239	AH3	0.2035
4	DH2	0.2191	AH1	0.1839
5	CO1	0.1911	DH1	0.1573
6	DH9	0.1055	DH7	0.0669
7	AH2	0.0988	CO5	0.0433
8	CO3	0.0852	CO3	0.0393
9	DH6	0.0314	CO1	0.0324
10	CO2	0.0249	DH3	0.0177
11	DH10	0	AH4	0.0079
12	DH7	-0.0011	AH2	0.0069
13	DH3	-0.0088	CO4	0.0059
14	CO6	-0.0228	DH10	0.0059
15	CO4	-0.0308	DH9	0
16	CO5	-0.0317	CO2	-0.0167
17	DH8	-0.0371	DH8	-0.0728
18	AH6	-0.0600	CO6	-0.0885
19	AH4	-0.0761	DH4	-0.0924
20	DH5	-0.0910	AH5	-0.1042
21	DH4	-0.1193	AH6	-0.1072
22	AH5	-0.1361	DH5	-0.1524

[7]The remaining 13 instances referred to either more than one piece or a class of pieces, thus were excluded in this experiment.

[8]www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[9]For the significant tests, we used McNemar test ($\alpha = 0.05$).

Table 5: Frequencies of REs relating to on-mouse

	pronouns	others	total
# all REs	548	693	1,241
# on-mouse	452	155	607
	(82.5%)	(22.4%)	(48.9%)

'# all REs' stands for the frequency of referring expressions uttered in the corpus and '# on-mouse' is the frequency of referring expressions in the situation when a referring expression is uttered and a mouse cursor is over the piece referred to by the expression.

feature of the pronoun and non-pronoun models. We calculate the weight of feature f shown in Table 2 according to the following formula.

$$\text{weight}(f) = \sum_{x \in SVs} w_x z_x(f) \qquad (1)$$

where *SVs* is a set of the support vectors in a ranker induced by *SVM*^{rank}, w_x is the weight of the support vector x, $z_x(f)$ is the function that returns 1 if f occurs in x, respectively.

The feature weights are shown in Table 4. This demonstrates that in the pronoun model the action history features have the highest weight, while with the non-pronoun model these features are less significant. As we can see in Table 5, pronouns are strongly related to the situation where a mouse cursor is over a piece, directly causing the weights of the features associated with the 'on-mouse' situation to become higher than other features.

On the other hand, in the non-pronoun model, the discourse history features, such as DH6 and DH2, are the most significant, indicating that the compatibility of the attributes of a piece and a referring expression is more crucial than other action history and current operation features. This is compatible with the previous research concerning textual reference resolution (Mitkov, 2002).

Table 4 shows that feature AH3 (aiming at capturing the recency in terms of a series of operations) is also significant. It empirically proves that the recent operation is strongly related to the salience of reference as a kind of 'focus' by humans.

5 Related Work

There have been increasing concerns about reference resolution in dialogue. Byron and Allen (1998) and Eckert and Strube (2000) reported about 50% of pronouns had no antecedent in TRAINS93 and Switchboard corpora respectively. Strube and Müller (2003) attempted to resolve pronominal anaphora in the Switchboard corpus by porting a corpus-based anaphora resolution model focusing on written texts (e.g. Soon et al. (2001) and Ng and Cardie (2002)). They used specialised features for spoken dialogues as well as conventional features. They reported relatively worse results than with written texts. The reason is that the features in their work capture only information derived from transcripts of dialogues, while it is also essential to bridge objects and concepts in the real (or virtual) world and their expressions (especially pronouns) for recognising referential relations intrinsically.

To improve performance on reference resolution in dialogue, researchers have focused on anaphoricity determination, which is the task of judging whether an expression explicitly has an antecedent in the text (i.e. in the preceding utterances) (Müller, 2006; Müller, 2007). Their work presented implementations of pronominal reference resolution in transcribed, multi-party dialogues. Müller (2006) focused on the determination of non-referential *it*, categorising instances of *it* in the ICSI Meeting Corpus (Janin et al., 2003) into six classes in terms of their grammatical categories. They also took into account each characteristic of these types by using a refined feature set. In the work by Müller (2007), they conducted an empirical evaluation including antecedent identification as well as anaphoricity determination. They used the relative frequencies of linguistic patterns as clues to introduce specific patterns for non-referentials. They reported that their performance for detecting non-referentials was relatively high (80.0% in precision and 60.9% in recall), while the overall performance was still low (18.2% in precision and 19.1% in recall). These results indicate the need for advancing research in reference resolution in dialogue.

In contrast to the above mentioned research, our task includes the treatment of entity disambiguation (i.e. selecting a referent out of a set of pieces on a computer display) as well as conventional anaphora resolution. Although our task setting is limited to the problem of solving the Tangram puzzle, we believe it is a good starting point for incorporating real (or virtual) world entities into coventional anaphora resolution.

6 Conclusion

This paper presented the task of reference resolution bridging pieces in the real world and their referents in dialogue. We presented an implementation of a reference resolution model exploiting extra-linguistic information, such as action history and current operation features, to capture the salience of operations by a participant and the arrangement of the pieces. Through our empirical evaluation, we demonstrated that the extra-linguistic information introduced in this paper contributed to improving performance. We also analysed the effect of each feature, showing that while action history features were useful for pronominal reference, discourse history features made sense for the other references.

In order to enhance this kind of reference resolution, there are several possible future directions. First, in the current problem setting, we exclude zero-anaphora (i.e. omitted expressions refer to either an expression in the previous utterances or an object on a display deictically). However, zero-anaphora is essential for precise modeling and recognition of reference because it is also directly related with the recency of referents, either textually or situationally. Second, representing distractors in a reference resolution model is also a key. Although, this paper presents an implementation of a reference model considering only the relationship between a referring expression and its candidate referents. However, there might be cases when the occurrence of expressions or manipulated pieces intervening between a referring expression and its referent need to be taken into account. Finally, more investigation is needed for considering other extra-linguistic information, such as eye-gaze, for exploring what kinds of information is critical to recognising reference in dialogue.

References

D. K. Byron and J. F. Allen. 1998. Resolving demonstrative pronouns in the trains93 corpus. In *Proceedings of the 2nd Colloquium on Discourse Anaphora and Anaphor Resolution* (*DAARC2*), pages 68–81.

D. K. Byron. 2005. Utilizing visual attention for cross-model coreference interpretation. In *CONTEXT 2005*, pages 83–96.

J. Carletta, R. L. Hill, C. Nicol, T. Taylor, J. P. de Ruiter, and E. G. Bard. 2010. Eyetracking for two-person tasks with manipulation of a virtual world. *Behavior Research Methods*, 42:254–265.

P. Denis and J. Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669.

B. P. W. Di Eugenio, R. H. Thomason, and J. D. Moore. 2000. The agreement process: An empirical investigation of human-human computer-mediated collaborative dialogues. *International Journal of Human-Computer Studies*, 53(6):1017–1076.

M. Eckert and M. Strube. 2000. Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89.

M. E. Foster, E. G. Bard, M. Guhe, R. L. Hill, J. Oberlander, and A. Knoll. 2008. The roles of haptic-ostensive referring expressions in cooperative, task-based human-robot dialogue. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction* (*HRI '08*), pages 295–302.

N. Ge, J. Hale, and E. Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 161–170.

B. J. Grosz, A. K. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the 10th EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30.

R. Iida, K. Inui, and Y. Matsumoto. 2005. Anaphora resolution by antecedent identification followed by anaphoricity determination. *ACM Transactions on Asian Language Information Processing* (*TALIP*), 4(4):417–434.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The ICSI meeting corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 364–367.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.

R. Mitkov. 2002. *Anaphora Resolution*. Studies in Language and Linguistics. Pearson Education.

C. Müller. 2006. Automatic detection of nonreferential *It* in spoken multi-party dialog. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–56.

C. Müller. 2007. Resolving *It*, *This*, and *That* in unrestricted multi-party dialog. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 816–823.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 104–111.

H. Poon and P. Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 650–659.

Z. Prasov and J. Y. Chai. 2008. What's in a gaze?: the role of eye-gaze in reference resolution in multimodal conversational interfaces. In *Proceedings of the 13th international conference on Intelligent user interfaces* (*IUI '08*), pages 20–29.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

P. Spanger, Y. Masaaki, R. Iida, and T. Takenobu. 2009. Using extra linguistic information for generating demonstrative pronouns in a situated collaboration task. In *Proceedings of Workshop on Production of Referring Expressions: Bridging the gap between computational and empirical approaches to reference*.

M. Strube and C. Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 168–175.

K. van Deemter. 2007. TUNA: Towards a unified algorithm for the generation of referring expressions. Technical report, Aberdeen University.

V. N. Vapnik. 1998. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing Communications, and control. John Wiley & Sons.

X. Yang, G. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 176–183.

X. Yang, J. Su, and C. L. Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceeding of the 43rd Annual Meeting of the Association for Computational Linguistics* (*ACL*), pages 165–172.

X. Yang, J. Su, J. Lang, C. L. Tan, T. Liu, and S. Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of Annual Meeting of the Association for Computational Linguistics* (*ACL*)*: Human Language Technologies* (*HLT*), pages 843–851.

Reading Between the Lines: Learning to Map High-level Instructions to Commands

S.R.K. Branavan, Luke S. Zettlemoyer, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{branavan, lsz, regina}@csail.mit.edu

Abstract

In this paper, we address the task of mapping *high-level instructions* to sequences of commands in an external environment. Processing these instructions is challenging—they posit goals to be achieved without specifying the steps required to complete them. We describe a method that fills in missing information using an automatically derived environment model that encodes states, transitions, and commands that cause these transitions to happen. We present an efficient approximate approach for learning this environment model as part of a policy-gradient reinforcement learning algorithm for text interpretation. This design enables learning for mapping high-level instructions, which previous statistical methods cannot handle.[1]

1 Introduction

In this paper, we introduce a novel method for mapping high-level instructions to commands in an external environment. These instructions specify goals to be achieved without explicitly stating all the required steps. For example, consider the first instruction in Figure 1 — "open control panel." The three GUI commands required for its successful execution are not explicitly described in the text, and need to be inferred by the user. This dependence on domain knowledge makes the automatic interpretation of high-level instructions particularly challenging.

The standard approach to this task is to start with both a manually-developed model of the environment, and rules for interpreting high-level instructions in the context of this model (Agre and Chapman, 1988; Di Eugenio and White, 1992; Di Eugenio, 1992; Webber et al., 1995). Given both the model and the rules, logic-based inference is used to automatically fill in the intermediate steps missing from the original instructions.

Our approach, in contrast, operates directly on the textual instructions in the context of the interactive environment, while requiring no additional information. By interacting with the environment and observing the resulting feedback, our method automatically learns both the mapping between the text and the commands, and the underlying model of the environment. One particularly noteworthy aspect of our solution is the interplay between the evolving mapping and the progressively acquired environment model as the system learns how to interpret the text. Recording the state transitions observed during interpretation allows the algorithm to construct a relevant model of the environment. At the same time, the environment model enables the algorithm to consider the consequences of commands before they are executed, thereby improving the accuracy of interpretation. Our method efficiently achieves both of these goals as part of a policy-gradient reinforcement learning algorithm.

We apply our method to the task of mapping software troubleshooting guides to GUI actions in the Windows environment (Branavan et al., 2009; Kushman et al., 2009). The key findings of our experiments are threefold. First, the algorithm can accurately interpret 61.5% of high-level instructions, which cannot be handled by previous statistical systems. Second, we demonstrate that explicitly modeling the environment also greatly improves the accuracy of processing low-level instructions, yielding a 14% absolute increase in performance over a competitive baseline (Branavan et al., 2009). Finally, we show the importance of constructing an environment model relevant to the language interpretation task — using textual

[1] Code, data, and annotations used in this work are available at http://groups.csail.mit.edu/rbg/code/rl-hli/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1268–1277,
Uppsala, Sweden, 11-16 July 2010.

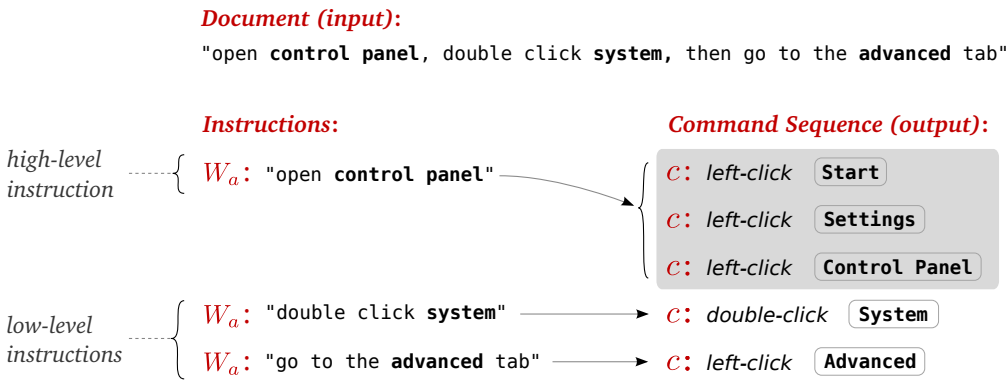


Figure 1: An example mapping of a document containing high-level instructions into a candidate sequence of five commands. The mapping process involves segmenting the document into individual instruction word spans W_a, and translating each instruction into the sequence $\vec{c}$ of one or more commands it describes. During learning, the correct output command sequence is not provided to the algorithm.

instructions enables us to bias exploration toward transitions relevant for language learning. This approach yields superior performance compared to a policy that relies on an environment model constructed via random exploration.

2 Related Work

Interpreting Instructions Our approach is most closely related to the reinforcement learning algorithm for mapping text instructions to commands developed by Branavan et al. (2009) (see Section 4 for more detail). Their method is predicated on the assumption that each command to be executed is explicitly specified in the instruction text. This assumption of a direct correspondence between the text and the environment is not unique to that paper, being inherent in other work on grounded language learning (Siskind, 2001; Oates, 2001; Yu and Ballard, 2004; Fleischman and Roy, 2005; Mooney, 2008; Liang et al., 2009; Matuszek et al., 2010). A notable exception is the approach of Eisenstein et al. (2009), which learns how an environment operates by reading text, rather than learning an explicit mapping from the text to the environment. For example, their method can learn the rules of a card game given instructions for how to play.

Many instances of work on instruction interpretation are replete with examples where instructions are formulated as high-level goals, targeted at users with relevant knowledge (Winograd, 1972; Di Eugenio, 1992; Webber et al., 1995; MacMahon et al., 2006). Not surprisingly, automatic approaches for processing such instructions have relied on hand-engineered world knowledge to reason about the preconditions and effects of environment commands. The assumption of a fully specified environment model is also common in work on semantics in the linguistics literature (Lascarides and Asher, 2004). While our approach learns to analyze instructions in a goal-directed manner, it does not require manual specification of relevant environment knowledge.

Reinforcement Learning Our work combines ideas of two traditionally disparate approaches to reinforcement learning (Sutton and Barto, 1998). The first approach, *model-based learning*, constructs a model of the environment in which the learner operates (e.g., modeling location, velocity, and acceleration in robot navigation). It then computes a policy directly from the rich information represented in the induced environment model. In the NLP literature, model-based reinforcement learning techniques are commonly used for dialog management (Singh et al., 2002; Lemon and Konstas, 2009; Schatzmann and Young, 2009). However, if the environment cannot be accurately approximated by a compact representation, these methods perform poorly (Boyan and Moore, 1995; Jong and Stone, 2007). Our instruction interpretation task falls into this latter category,[2] rendering standard model-based learning ineffective.

The second approach – model-free methods such as *policy learning* – aims to select the opti-

[2]For example, in the Windows GUI domain, clicking on the *File* menu will result in a different submenu depending on the application. Thus it is impossible to predict the effects of a previously unseen GUI command.

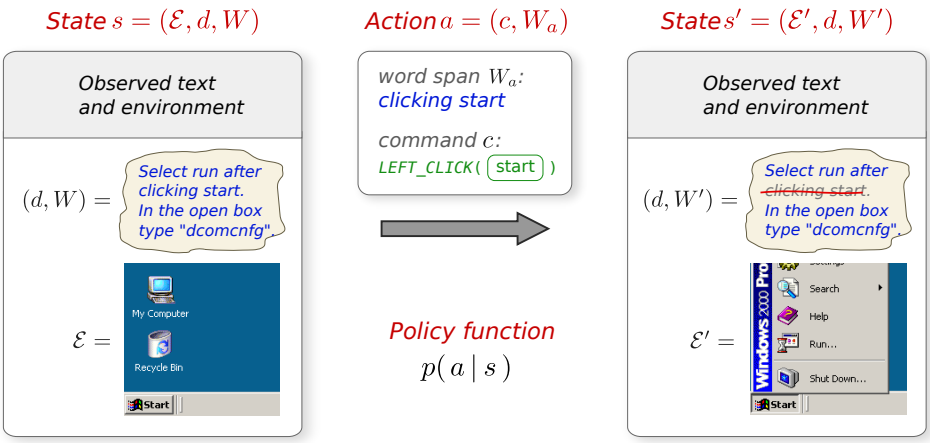


Figure 2: A single step in the instruction mapping process formalized as an MDP. State s is comprised of the state of the external environment $\mathcal{E}$, and the state of the document (d, W), where W is the list of all word spans mapped by previous actions. An action a selects a span W_a of unused words from (d, W), and maps them to an environment command c. As a consequence of a, the environment state changes to $\mathcal{E}' \sim p(\mathcal{E}'|\mathcal{E}, c)$, and the list of mapped words is updated to $W' = W \cup W_a$.

mal action at every step, without explicitly constructing a model of the environment. While policy learners can effectively operate in complex environments, they are not designed to benefit from a learned environment model. We address this limitation by expanding a policy learning algorithm to take advantage of a partial environment model estimated during learning. The approach of conditioning the policy function on future reachable states is similar in concept to the use of *post-decision state* information in the approximate dynamic programming framework (Powell, 2007).

3 Problem Formulation

Our goal is to map instructions expressed in a natural language document d into the corresponding sequence of commands $\vec{c} = \langle c_1, \ldots, c_m \rangle$ executable in an environment. As input, we are given a set of raw instruction documents, an environment, and a reward function as described below.

The *environment* is formalized as its states and transition function. An *environment state* $\mathcal{E}$ specifies the objects accessible in the environment at a given time step, along with the objects' properties. The environment state transition function $p(\mathcal{E}'|\mathcal{E}, c)$ encodes how the state changes from $\mathcal{E}$ to $\mathcal{E}'$ in response to a command c.[3] During learning, this function is not known, but samples from it can be collected by executing commands and observing the resulting environment state. A real-valued *reward function* measures how well a command sequence $\vec{c}$ achieves the task described in the document.

We posit that a document d is composed of a sequence of instructions, each of which can take one of two forms:

- *Low-level instructions*: these explicitly describe single commands.[4] E.g., "double click system" in Figure 1.
- *High-level instructions*: these correspond to a sequence of one or more environment commands, none of which are explicitly described by the instruction. E.g., "open control panel" in Figure 1.

4 Background

Our innovation takes place within a previously established general framework for the task of mapping instructions to commands (Branavan et al., 2009). This framework formalizes the mapping process as a *Markov Decision Process* (MDP) (Sutton and Barto, 1998), with actions encoding individual instruction-to-command mappings, and states representing partial interpretations of the document. In this section, we review the details of this framework.

[3] While in the general case the environment state transitions maybe stochastic, they are deterministic in the software GUI used in this work.

[4] Previous work (Branavan et al., 2009) is only able to handle low-level instructions.

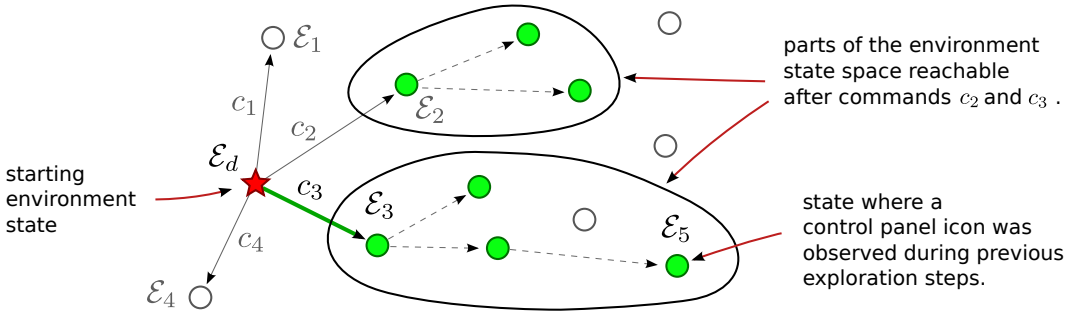


Figure 3: Using information derived from future states to interpret the high-level instruction "open control panel." $\mathcal{E}_d$ is the starting state, and c_1 through c_4 are candidate commands. Environment states are shown as circles, with previously visited environment states colored green. Dotted arrows show known state transitions. All else being equal, the information that the control panel icon was observed in state $\mathcal{E}_5$ during previous exploration steps can help to correctly select command c_3.

States and Actions A document is interpreted by incrementally constructing a sequence of actions. Each action selects a word span from the document, and maps it to one environment command. To predict actions sequentially, we track the states of the environment and the document over time as shown in Figure 2. This *mapping state* s is a tuple $(\mathcal{E}, d, W)$ where $\mathcal{E}$ is the current environment state, d is the document being interpreted, and W is the list of word spans selected by previous actions. The mapping state s is observed prior to selecting each action.

The *mapping action* a is a tuple (c, W_a) that represents the joint selection of a span of words W_a and an environment command c. Some of the candidate actions would correspond to the correct instruction mappings, e.g., (c = *double-click* system, W_a = "double click system"). Others such as (c = *left-click* system, W_a = "double click system") would be erroneous. The algorithm learns to interpret instructions by learning to construct sequences of actions that assign the correct commands to the words.

The interpretation of a document d begins at an initial mapping state $s_0 = (\mathcal{E}_d, d, \emptyset)$, $\mathcal{E}_d$ being the starting state of the environment for the document. Given a state $s = (\mathcal{E}, d, W)$, the space of possible actions $a = (c, W_a)$ is defined by enumerating sub-spans of unused words in d and candidate commands in $\mathcal{E}$.[5] The action to execute, a, is selected based on a *policy function* $p(a|s)$ by finding $\arg\max_a p(a|s)$. Performing action a in state $s = (\mathcal{E}, d, W)$ results in a new state s' according to the distribution $p(s'|s, a)$, where:

$$\begin{aligned} a &= (c, W_a), \\ \mathcal{E}' &\sim p(\mathcal{E}'|\mathcal{E}, c), \\ W' &= W \cup W_a, \\ s' &= (\mathcal{E}', d, W'). \end{aligned}$$

The process of selecting and executing actions is repeated until all the words in d have been mapped.[6]

A Log-Linear Parameterization The policy function used for action selection is defined as a log-linear distribution over actions:

$$p(a|s;\theta) = \frac{e^{\theta \cdot \phi(s,a)}}{\sum_{a'} e^{\theta \cdot \phi(s,a')}}, \quad (1)$$

where $\theta \in \mathbb{R}^n$ is a weight vector, and $\phi(s, a) \in \mathbb{R}^n$ is an n-dimensional feature function. This representation has the flexibility to incorporate a variety of features computed on the states and actions.

Reinforcement Learning Parameters of the policy function $p(a|s;\theta)$ are estimated to maximize the expected future reward for analyzing each document $d \in D$:

$$\theta = \arg\max_\theta \; E_{p(h|\theta)}\left[r(h)\right], \quad (2)$$

where $h = (s_0, a_0, \ldots, s_{m-1}, a_{m-1}, s_m)$ is a *history* that records the analysis of document d, $p(h|\theta)$ is the probability of selecting this analysis given policy parameters θ, and the reward $r(h)$ is a real valued indication of the quality of h.

[5] Here, command reordering is possible. At each step, the span of selected words W_a is not required to be adjacent to the previous selections. This reordering is used to interpret sentences such as "Select exit after opening the File menu."

[6] To account for document words that are not part of an instruction, c may be a *null* command.

5 Algorithm

We expand the scope of learning approaches for automatic document interpretation by enabling the analysis of *high-level instructions*. The main challenge in processing these instructions is that, in contrast to their low-level counterparts, they correspond to sequences of one or more commands. A simple way to enable this one-to-many mapping is to allow actions that do not consume words (i.e., $|W_a| = 0$). The sequence of actions can then be constructed incrementally using the algorithm described above. However, this change significantly complicates the interpretation problem – we need to be able to predict commands that are not directly described by any words, and allowing action sequences significantly increases the space of possibilities for each instruction. Since we cannot enumerate all possible sequences at decision time, we limit the space of possibilities by learning which sequences are likely to be relevant for the current instruction.

To motivate the approach, consider the decision problem in Figure 3, where we need to find a command sequence for the high-level instruction "open control panel." The algorithm focuses on command sequences leading to environment states where the control panel icon was previously observed. The information about such states is acquired during exploration and is stored in a *partial environment model* $q(\mathcal{E}'|\mathcal{E}, c)$.

Our goal is to map high-level instructions to command sequences by leveraging knowledge about the long-term effects of commands. We do this by integrating the partial environment model into the policy function. Specifically, we modify the log-linear policy $p(a|s; q, \theta)$ by adding *look-ahead features* $\phi(s, a, q)$ which complement the local features used in the previous model. These look-ahead features incorporate various measurements that characterize the potential of future states reachable via the selected action. Although primarily designed to analyze high-level instructions, this approach is also useful for mapping low-level instructions.

Below, we first describe how we estimate the partial environment transition model and how this model is used to compute the look-ahead features. This is followed by the details of parameter estimation for our algorithm.

5.1 Partial Environment Transition Model

To compute the look-ahead features, we first need to collect statistics about the environment transition function $p(\mathcal{E}'|\mathcal{E}, c)$. An example of an environment transition is the change caused by clicking on the "start" button. We collect this information through observation, and build a partial environment transition model $q(\mathcal{E}'|\mathcal{E}, c)$.

One possible strategy for constructing q is to observe the effects of executing random commands in the environment. In a complex environment, however, such a strategy is unlikely to produce state samples relevant to our text analysis task. Instead, we use the training documents to guide the sampling process. During training, we execute the command sequences predicted by the policy function in the environment, caching the resulting state transitions. Initially, these commands may have little connection to the actual instructions. As learning progresses and the quality of the interpretation improves, more promising parts of the environment will be observed. This process yields samples that are biased toward the content of the documents.

5.2 Look-Ahead Features

We wish to select actions that allow for the best follow-up actions, thereby finding the analysis with the highest total reward for a given document. In practice, however, we do not have information about the effects of all possible future actions. Instead, we capitalize on the state transitions observed during the sampling process described above, allowing us to incrementally build an environment model of actions and their effects.

Based on this transition information, we can estimate the usefulness of actions by considering the properties of states they can reach. For instance, some states might have very low immediate reward, indicating that they are unlikely to be part of the best analysis for the document. While the usefulness of most states is hard to determine, it correlates with various properties of the state. We encode the following properties as look-ahead features in our policy:

- The highest reward achievable by an action sequence passing through this state. This property is computed using the learned environment model, and is therefore an approximation.

- The length of the above action sequence.
- The average reward received at the environment state while interpreting any document. This property introduces a bias towards commonly visited states that frequently recur throughout multiple documents' correct interpretations.

Because we can never encounter all states and all actions, our environment model is always incomplete and these properties can only be computed based on partial information. Moreover, the predictive strength of the properties is not known in advance. Therefore we incorporate them as separate features in the model, and allow the learning process to estimate their weights. In particular, we select actions a based on the current state s and the partial environment model q, resulting in the following policy definition:

$$p(a|s;q,\theta) = \frac{e^{\theta \cdot \phi(s,a,q)}}{\sum_{a'} e^{\theta \cdot \phi(s,a',q)}}, \qquad (3)$$

where the feature representation $\phi(s,a,q)$ has been extended to be a function of q.

5.3 Parameter Estimation

The learning algorithm is provided with a set of documents $d \in D$, an environment in which to execute command sequences $\vec{c}$, and a reward function $r(h)$. The goal is to estimate two sets of parameters: 1) the parameters θ of the policy function, and 2) the partial environment transition model $q(\mathcal{E}'|\mathcal{E},c)$, which is the observed portion of the true model $p(\mathcal{E}'|\mathcal{E},c)$. These parameters are mutually dependent: θ is defined over a feature space dependent on q, and q is sampled according to the policy function parameterized by θ.

Algorithm 1 shows the procedure for joint learning of these parameters. As in standard policy gradient learning (Sutton et al., 2000), the algorithm iterates over all documents $d \in D$ (steps 1, 2), selecting and executing actions in the environment (steps 3 to 6). The resulting reward is used to update the parameters θ (steps 8, 9). In the new joint learning setting, this process also yields samples of state transitions which are used to estimate $q(\mathcal{E}'|\mathcal{E},c)$ (step 7). This updated q is then used to compute the feature functions $\phi(s,a,q)$ during the next iteration of learning (step 4). This process is repeated until the total reward on training documents converges.

Input: A document set D,
Feature function ϕ,
Reward function $r(h)$,
Number of iterations T

Initialization: Set θ to small random values.
Set q to the empty set.

1 **for** $i = 1 \cdots T$ **do**
2 **foreach** $d \in D$ **do**

Sample history $h \sim p(h|\theta)$ where $h = (s_0, a_0, \cdots, a_{n-1}, s_n)$ as follows:

Initialize environment to document specific starting state $\mathcal{E}_d$

3 **for** $t = 0 \cdots n-1$ **do**
4 Compute $\phi(a, s_t, q)$ based on latest q
5 Sample action $a_t \sim p(a|s_t;q,\theta)$
6 Execute a_t on state s_t: $s_{t+1} \sim p(s|s_t,a_t)$
7 Set $q = q \cup \{(\mathcal{E}', \mathcal{E}, c)\}$ where $\mathcal{E}'$, $\mathcal{E}$, c are the environment states and commands from s_{t+1}, s_t, and a_t
end
8 $\Delta \leftarrow$
$$\sum_t \left[\phi(s_t, a_t, q) - \sum_{a'} \phi(s_t, a', q)\, p(a'|s_t;q,\theta) \right]$$
9 $\theta \leftarrow \theta + r(h)\Delta$
end
end

Output: Estimate of parameters θ

Algorithm 1: A policy gradient algorithm that also learns a model of the environment.

This algorithm capitalizes on the synergy between θ and q. As learning proceeds, the method discovers a more complete state transition function q, which improves the accuracy of the look-ahead features, and ultimately, the quality of the resulting policy. An improved policy function in turn produces state samples that are more relevant to the document interpretation task.

6 Applying the Model

We apply our algorithm to the task of interpreting help documents to perform software related tasks (Branavan et al., 2009; Kushman et al., 2009). Specifically, we consider documents from Microsoft's Help and Support website.[7] As in prior work, we use a virtual machine set-up to allow our method to interact with a Windows 2000 environment.

Environment States and Actions In this application of our model, the environment state is the set of visible user interface (UI) objects, along

[7] http://support.microsoft.com/

with their properties (e.g., the object's label, parent window, etc). The environment commands consist of the UI commands *left-click*, *right-click*, *double-click*, and *type-into*. Each of these commands requires a UI object as a parameter, while *type-into* needs an additional parameter containing the text to be typed. On average, at each step of the interpretation process, the branching factor is 27.14 commands.

Reward Function An ideal reward function would be to verify whether the task specified by the help document was correctly completed. Since such verification is a challenging task, we rely on a noisy approximation: we assume that each sentence specifies at least one command, and that the text describing the command has words matching the label of the environment object. If a history h has at least one such command for each sentence, the environment reward function $r(h)$ returns a positive value, otherwise it returns a negative value. This environment reward function is a simplification of the one described in Branavan et al. (2009), and it performs comparably in our experiments.

Features In addition to the look-ahead features described in Section 5.2, the policy also includes the set of features used by Branavan et al. (2009). These features are functions of both the text and environment state, modeling local properties that are useful for action selection.

7 Experimental Setup

Datasets Our model is trained on the same dataset used by Branavan et al. (2009). For testing we use two datasets: the first one was used in prior work and contains only low-level instructions, while the second dataset is comprised of documents with high-level instructions. This new dataset was collected from the Microsoft Help and Support website, and has on average 1.03 high-level instructions per document. The second dataset contains 60 test documents, while the first is split into 70, 18 and 40 document for training, development and testing respectively. The combined statistics for these datasets is shown below:

Total # of documents	188
Total # of words	7448
Vocabulary size	739
Avg. actions per document	10

Reinforcement Learning Parameters Following common practice, we encourage exploration during learning with an ϵ-greedy strategy (Sutton and Barto, 1998), with ϵ set to 0.1. We also identify *dead-end* states, i.e. states with the lowest possible immediate reward, and use the induced environment model to encourage additional exploration by lowering the likelihood of actions that lead to such dead-end states.

During the early stages of learning, experience gathered in the environment model is extremely sparse, causing the look-ahead features to provide poor estimates. To speed convergence, we ignore these estimates by disabling the look-ahead features for a fixed number of initial training iterations.

Finally, to guarantee convergence, stochastic gradient ascent algorithms require a learning rate schedule. We use a modified *search-then-converge* algorithm (Darken and Moody, 1990), and tie the learning rate to the ratio of training documents that received a positive reward in the current iteration.

Baselines As a baseline, we compare our method against the results reported by Branavan et al. (2009), denoted here as BCZB09.

As an upper bound for model performance, we also evaluate our method using a reward signal that simulates a fully-supervised training regime. We define a reward function that returns positive one for histories that match the annotations, and zero otherwise. Performing policy-gradient with this function is equivalent to training a fully-supervised, stochastic gradient algorithm that optimizes conditional likelihood (Branavan et al., 2009).

Evaluation Metrics We evaluate the accuracy of the generated mapping by comparing it against manual annotations of the correct action sequences. We measure the percentage of correct actions and the percentage of documents where every action is correct. In general, the sequential nature of the interpretation task makes it difficult to achieve high action accuracy. For example, executing an incorrect action early on, often leads to an environment state from which the remaining instructions cannot be completed. When this happens, it is not possible to recover the remaining actions, causing cascading errors that significantly reduce performance.

	Low-level instruction dataset		High-level instruction dataset		
	action	document	action	high-level action	document
BCZB09	0.647	0.375	0.021	0.022	0.000
BCZB09 + annotation	$*$ 0.756	0.525	0.035	0.022	0.000
Our model	**0.793**	**0.517**	$*$ **0.419**	$*$ **0.615**	$*$ **0.283**
Our model + annotation	0.793	0.650	$*$ 0.357	0.492	0.333

Table 1: Accuracy of the mapping produced by our model, its variants, and the baseline. Values marked with $*$ are statistically significant at $p < 0.01$ compared to the value immediately above it.

8 Results

As shown in Table 1, our model outperforms the baseline on the two datasets, according to all evaluation metrics. In contrast to the baseline, our model can handle high-level instructions, accurately interpreting 62% of them in the second dataset. Every document in this set contains at least one high-level action, which on average, maps to 3.11 environment commands each. The overall action performance on this dataset, however, seems unexpectedly low at 42%. This discrepancy is explained by the fact that in this dataset, high-level instructions are often located towards the beginning of the document. If these initial challenging instructions are not processed correctly, the rest of the actions for the document cannot be interpreted.

As the performance on the first dataset indicates, the new algorithm is also beneficial for processing low-level instructions. The model outperforms the baseline by at least 14%, both in terms of the actions and the documents it can process. Not surprisingly, the best performance is achieved when the new algorithm has access to manually annotated data during training.

We also performed experiments to validate the intuition that the partial environment model must contain information relevant for the language interpretation task. To test this hypothesis, we replaced the learned environment model with one of the same size gathered by executing random commands. The model with randomly sampled environment transitions performs poorly: it can only process 4.6% of documents and 15% of actions on the dataset with high-level instructions, compared to 28.3% and 41.9% respectively for our algorithm. This result also explains why training with full supervision hurts performance on high-level instructions (see Table 1). Learning directly from annotations results in a low-quality environment model due to the relative lack of exploration, hurting the model's ability to leverage the look-ahead features.

High-level instruction
- open **device manager**

Extracted low-level instruction paraphrase
- double click **my computer**
- double click **control panel**
- double click **administrative tools**
- double click **computer management**
- double click **device manager**

High-level instruction
- open the **network tool** in **control panel**

Extracted low-level instruction paraphrase
- click **start**
- point to **settings**
- click **control panel**
- double click **network and dial-up connections**

Figure 4: Examples of automatically generated paraphrases for high-level instructions. The model maps the high-level instruction into a sequence of commands, and then translates them into the corresponding low-level instructions.

Finally, to demonstrate the quality of the learned word–command alignments, we evaluate our method's ability to paraphrase from high-level instructions to low-level instructions. Here, the goal is to take each high-level instruction and construct a text description of the steps required to achieve it. We did this by finding high-level instructions where each of the commands they are associated with is also described by a low-level instruction in some other document. For example, if the text "open control panel" was mapped to the three commands in Figure 1, and each of those commands was described by a low-level instruction elsewhere, this procedure would create a paraphrase such as "click start, left click setting, and select control panel." Of the 60 high-level instructions tagged in the test set, this approach found paraphrases for 33 of them. 29 of

these paraphrases were correct, in the sense that they describe all the necessary commands. Figure 4 shows some examples of the automatically extracted paraphrases.

9 Conclusions and Future Work

In this paper, we demonstrate that knowledge about the environment can be learned and used effectively for the task of mapping instructions to actions. A key feature of this approach is the synergy between language analysis and the construction of the environment model: instruction text drives the sampling of the environment transitions, while the acquired environment model facilitates language interpretation. This design enables us to learn to map high-level instructions while also improving accuracy on low-level instructions.

To apply the above method to process a broad range of natural language documents, we need to handle several important semantic and pragmatic phenomena, such as reference, quantification, and conditional statements. These linguistic constructions are known to be challenging to learn – existing approaches commonly rely on large amounts of hand annotated data for training. An interesting avenue of future work is to explore an alternative approach which learns these phenomena by combining linguistic information with knowledge gleaned from an automatically induced environment model.

Acknowledgments

The authors acknowledge the support of the NSF (CAREER grant IIS-0448168, grant IIS-0835445, and grant IIS-0835652) and the Microsoft Research New Faculty Fellowship. Thanks to Aria Haghighi, Leslie Pack Kaelbling, Tom Kwiatkowski, Martin Rinard, David Silver, Mark Steedman, Csaba Szepesvari, the MIT NLP group, and the ACL reviewers for their suggestions and comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Philip E. Agre and David Chapman. 1988. What are plans for? Technical report, Cambridge, MA, USA.

J. A. Boyan and A. W. Moore. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in NIPS*, pages 369–376.

S.R.K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of ACL*, pages 82–90.

Christian Darken and John Moody. 1990. Note on learning rate schedules for stochastic optimization. In *Advances in NIPS*, pages 832–838.

Barbara Di Eugenio and Michael White. 1992. On the interpretation of natural language instructions. In *Proceedings of COLING*, pages 1147–1151.

Barbara Di Eugenio. 1992. Understanding natural language instructions: the case of purpose clauses. In *Proceedings of ACL*, pages 120–127.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of EMNLP*, pages 958–967.

Michael Fleischman and Deb Roy. 2005. Intentional context in situated natural language learning. In *Proceedings of CoNLL*, pages 104–111.

Nicholas K. Jong and Peter Stone. 2007. Model-based function approximation in reinforcement learning. In *Proceedings of AAMAS*, pages 670–677.

Nate Kushman, Micah Brodsky, S.R.K. Branavan, Dina Katabi, Regina Barzilay, and Martin Rinard. 2009. Wikido. In *Proceedings of HotNets-VIII*.

Alex Lascarides and Nicholas Asher. 2004. Imperatives in dialogue. In P. Kuehnlein, H. Rieser, and H. Zeevat, editors, *The Semantics and Pragmatics of Dialogue for the New Millenium*. Benjamins.

Oliver Lemon and Ioannis Konstas. 2009. User simulations for context-sensitive speech recognition in spoken dialogue systems. In *Proceedings of EACL*, pages 505–513.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of ACL*, pages 91–99.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of AAAI*, pages 1475–1482.

C. Matuszek, D. Fox, and K. Koscher. 2010. Following directions using statistical machine translation. In *Proceedings of Human-Robot Interaction*, pages 251–258.

Raymond J. Mooney. 2008. Learning to connect language and perception. In *Proceedings of AAAI*, pages 1598–1601.

James Timothy Oates. 2001. *Grounding knowledge in sensors: Unsupervised learning for language and planning*. Ph.D. thesis, University of Massachusetts Amherst.

Warren B Powell. 2007. *Approximate Dynamic Programming*. Wiley-Interscience.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE Trans. Audio, Speech and Language Processing*, 17(4):733–747.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Jeffrey Mark Siskind. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in NIPS*, pages 1057–1063.

Bonnie Webber, Norman Badler, Barbara Di Eugenio, Libby Levison Chris Geib, and Michael Moore. 1995. Instructions, intentions and expectations. *Artificial Intelligence*, 73(1-2).

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of AAAI*, pages 488–493.

Profiting from Mark-Up: Hyper-Text Annotations for Guided Parsing

Valentin I. Spitkovsky
Computer Science Department
Stanford University and Google Inc.
valentin@google.com

Daniel Jurafsky
Departments of Linguistics and
Computer Science, Stanford University
jurafsky@stanford.edu

Hiyan Alshawi
Google Inc.
hiyan@google.com

Abstract

We show how web mark-up can be used to improve unsupervised dependency parsing. Starting from raw bracketings of four common HTML tags (anchors, bold, italics and underlines), we refine approximate partial phrase boundaries to yield accurate parsing constraints. Conversion procedures fall out of our linguistic analysis of a newly available million-word hyper-text corpus. We demonstrate that derived constraints aid grammar induction by training Klein and Manning's Dependency Model with Valence (DMV) on this data set: parsing accuracy on Section 23 (all sentences) of the Wall Street Journal corpus jumps to 50.4%, beating previous state-of-the-art by more than 5%. Web-scale experiments show that the DMV, perhaps because it is unlexicalized, does not benefit from orders of magnitude more annotated but noisier data. Our model, trained on a single blog, generalizes to 53.3% accuracy out-of-domain, against the Brown corpus — nearly 10% higher than the previous published best. The fact that web mark-up strongly correlates with syntactic structure may have broad applicability in NLP.

1 Introduction

Unsupervised learning of hierarchical syntactic structure from free-form natural language text is a hard problem whose eventual solution promises to benefit applications ranging from question answering to speech recognition and machine translation. A restricted version of this problem that targets dependencies and assumes partial annotation — sentence boundaries and part-of-speech (POS) tagging — has received much attention. Klein and Manning (2004) were the first to beat a simple parsing heuristic, the right-branching baseline; today's state-of-the-art systems (Headden et al., 2009; Cohen and Smith, 2009; Spitkovsky et al., 2010a) are rooted in their Dependency Model with Valence (DMV), still trained using variants of EM.

Pereira and Schabes (1992) outlined three major problems with classic EM, applied to a related problem, constituent parsing. They extended classic inside-outside re-estimation (Baker, 1979) to respect any bracketing constraints included with a training corpus. This conditioning on partial parses addressed all three problems, leading to: (i) linguistically reasonable constituent boundaries and induced grammars more likely to agree with qualitative judgments of sentence structure, which is underdetermined by unannotated text; (ii) fewer iterations needed to reach a good grammar, countering convergence properties that sharply deteriorate with the number of non-terminal symbols, due to a proliferation of local maxima; and (iii) better (in the best case, linear) time complexity per iteration, versus running time that is ordinarily cubic in both sentence length *and* the total number of non-terminals, rendering sufficiently large grammars computationally impractical. Their algorithm sometimes found good solutions from bracketed corpora but not from raw text, supporting the view that purely unsupervised, self-organizing inference methods can miss the trees for the forest of distributional regularities. This was a promising break-through, but the problem of whence to get partial bracketings was left open.

We suggest mining partial bracketings from a cheap and abundant natural language resource: the hyper-text mark-up that annotates web-pages. For example, consider that anchor text can match linguistic constituents, such as verb phrases, exactly:

..., whereas McCain is secure on the topic, Obama <a>[VP worries about winning the pro-Israel vote].

To validate this idea, we created a new data set, novel in combining a real blog's raw HTML with tree-bank-like constituent structure parses, gener-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1278–1287,
Uppsala, Sweden, 11-16 July 2010.

ated automatically. Our linguistic analysis of the most prevalent tags (anchors, bold, italics and underlines) over its 1M$^+$ words reveals a strong connection between syntax and mark-up (all of our examples draw from this corpus), inspiring several simple techniques for automatically deriving parsing constraints. Experiments with both hard and more flexible constraints, as well as with different styles and quantities of annotated training data — the blog, web news and the web itself, confirm that mark-up-induced constraints consistently improve (otherwise unsupervised) dependency parsing.

2 Intuition and Motivating Examples

It is natural to expect hidden structure to seep through when a person annotates a sentence. As it happens, a non-trivial fraction of the world's population routinely annotates text diligently, if only partially and informally.[1] They inject hyper-links, vary font sizes, and toggle colors and styles, using mark-up technologies such as HTML and XML.

As noted, web annotations can be indicative of phrase boundaries, e.g., in a complicated sentence:

> In 1998, however, as I <a>[VP established in [NP *The New Republic*] and Bill Clinton just <a>[VP confirmed in his memoirs], Netanyahu changed his mind and ...

In doing so, mark-up sometimes offers useful cues even for low-level tokenization decisions:

> [NP [NP Libyan ruler]
> <a>[NP Mu'ammar al-Qaddafi] referred to ...

```
(NP (ADJP (NP (JJ Libyan) (NN ruler))
          (JJ Mu))
    (`` `) (NN ammar) (NNS al-Qaddafi))
```

Above, a backward quote in an Arabic name confuses the Stanford parser.[2] Yet mark-up lines up with the broken noun phrase, signals cohesion, and moreover sheds light on the internal structure of a compound. As Vadas and Curran (2007) point out, such details are frequently omitted even from manually compiled tree-banks that err on the side of flat annotations of base-NPs.

Admittedly, not all boundaries between HTML tags and syntactic constituents match up nicely:

> ..., but [S [NP the <a>*Toronto Star*][VP reports [NP this][PP in the softest possible way],[S stating only that ...]]]

Combining parsing with mark-up may not be straight-forward, but there is hope: even above, one of each nested tag's boundaries aligns; and *Toronto Star*'s neglected determiner could be forgiven, certainly within a dependency formulation.

3 A High-Level Outline of Our Approach

Our idea is to implement the DMV (Klein and Manning, 2004) — a standard unsupervised grammar inducer. But instead of learning the unannotated test set, we train with text that contains web mark-up, using various ways of converting HTML into parsing constraints. We still test on WSJ (Marcus et al., 1993), in the standard way, and also check generalization against a hidden data set — the Brown corpus (Francis and Kucera, 1979). Our parsing constraints come from a blog — a new corpus we created, the web and news (see Table 1 for corpora's sentence and token counts).

To facilitate future work, we make the final models and our manually-constructed blog data publicly available.[3] Although we are unable to share larger-scale resources, our main results should be reproducible, as both linguistic analysis and our best model rely exclusively on the blog.

Corpus	*Sentences*	*POS Tokens*
WSJ$^\infty$	49,208	1,028,347
Section 23	2,353	48,201
WSJ45	48,418	986,830
WSJ15	15,922	163,715
Brown100	24,208	391,796
BLOG$_p$	57,809	1,136,659
BLOG$_t$45	56,191	1,048,404
BLOG$_t$15	23,214	212,872
NEWS45	2,263,563,078	32,119,123,561
NEWS15	1,433,779,438	11,786,164,503
WEB45	8,903,458,234	87,269,385,640
WEB15	7,488,669,239	55,014,582,024

Table 1: Sizes of corpora derived from WSJ and Brown, as well as those we collected from the web.

4 Data Sets for Evaluation and Training

The appeal of unsupervised parsing lies in its ability to learn from surface text alone; but (intrinsic) evaluation still requires parsed sentences. Following Klein and Manning (2004), we begin with reference constituent parses and compare against deterministically derived dependencies: after pruning out all empty subtrees, punctuation and terminals (tagged # and $) not pronounced where they appear, we drop all sentences with more than a prescribed number of tokens remaining and use automatic "head-percolation" rules (Collins, 1999) to convert the rest, as is standard practice.

[1] Even when (American) grammar schools lived up to their name, they only taught dependencies. This was back in the days before constituent grammars were invented.

[2] http://nlp.stanford.edu:8080/parser/

[3] http://cs.stanford.edu/~valentin/

Length Cutoff	*Marked Sentences*	*POS Tokens*	*Bracketings*	
			All	*Multi-Token*
0	6,047	1,136,659	7,731	6,015
1	*of 57,809*	149,483	7,731	6,015
2	4,934	124,527	6,482	6,015
3	3,295	85,423	4,476	4,212
4	2,103	56,390	2,952	2,789
5	1,402	38,265	1,988	1,874
6	960	27,285	1,365	1,302
7	692	19,894	992	952
8	485	14,528	710	684
9	333	10,484	499	479
10	245	7,887	365	352
15	42	1,519	65	63
20	13	466	20	20
25	6	235	10	10
30	3	136	6	6
40	0	0	0	0

Table 2: Counts of sentences, tokens and (unique) bracketings for BLOG_p, restricted to only those sentences having at least one bracketing no shorter than the length cutoff (but shorter than the sentence).

Our primary reference sets are derived from the Penn English Treebank's Wall Street Journal portion (Marcus et al., 1993): WSJ45 (sentences with fewer than 46 tokens) and Section 23 of WSJ^{∞} (all sentence lengths). We also evaluate on Brown100, similarly derived from the parsed portion of the Brown corpus (Francis and Kucera, 1979). While we use WSJ45 and WSJ15 to train baseline models, the bulk of our experiments is with web data.

4.1 A News-Style Blog: Daniel Pipes

Since there was no corpus overlaying syntactic structure with mark-up, we began constructing a new one by downloading articles[4] from a news-style blog. Although limited to a single genre — political opinion, `danielpipes.org` is clean, consistently formatted, carefully edited and larger than WSJ (see Table 1). Spanning decades, Pipes' editorials are mostly in-domain for POS taggers and tree-bank-trained parsers; his recent (internet-era) entries are thoroughly cross-referenced, conveniently providing just the mark-up we hoped to study via uncluttered (printer-friendly) HTML.[5]

After extracting moderately clean text and mark-up locations, we used MxTerminator (Reynar and Ratnaparkhi, 1997) to detect sentence boundaries. This initial automated pass begot multiple rounds of various semi-automated clean-ups that involved fixing sentence breaking, modifying parser-unfriendly tokens, converting HTML entities and non-ASCII text, correcting typos, and so on. After throwing away annotations of fractional words (e.g., *basmachi*s) and tokens (e.g., *Sesame Street*-like), we broke up all mark-up that crossed sentence boundaries (i.e., loosely speaking, replaced constructs like <u>...]\[s...</u> with <u>...</u>]\[s <u>...</u>) and discarded any tags left covering entire sentences.

We finalized two versions of the data: BLOG_t, tagged with the Stanford tagger (Toutanova and Manning, 2000; Toutanova et al., 2003),[6] and BLOG_p, parsed with Charniak's parser (Charniak, 2001; Charniak and Johnson, 2005).[7] The reason for this dichotomy was to use state-of-the-art parses to analyze the relationship between syntax and mark-up, yet to prevent jointly tagged (and non-standard `AUX[G]`) POS sequences from interfering with our (otherwise unsupervised) training.[8]

4.2 Scaled up *Quantity*: The (English) Web

We built a large (see Table 1) but messy data set, WEB — English-looking web-pages, pre-crawled by a search engine. To avoid machine-generated spam, we excluded low quality sites flagged by the indexing system. We kept only sentence-like runs of words (satisfying punctuation and capitalization constraints), POS-tagged with TnT (Brants, 2000).

4.3 Scaled up *Quality*: (English) Web News

In an effort to trade quantity for quality, we constructed a smaller, potentially cleaner data set, NEWS. We reckoned editorialized content would lead to fewer extracted non-sentences. Perhaps surprisingly, NEWS is less than an order of magnitude smaller than WEB (see Table 1); in part, this is due to less aggressive filtering — we trust sites approved by the human editors at Google News.[9] In all other respects, our pre-processing of NEWS pages was identical to our handling of WEB data.

[4] `http://danielpipes.org/art/year/all`

[5] `http://danielpipes.org/article_print.php?id=...`

[6] `http://nlp.stanford.edu/software/stanford-postagger-2008-09-28.tar.gz`

[7] `ftp://ftp.cs.brown.edu/pub/nlparser/parser05Aug16.tar.gz`

[8] However, since many taggers are themselves trained on manually parsed corpora, such as WSJ, no parser that relies on external POS tags could be considered truly unsupervised; for a *fully* unsupervised example, see Seginer's (2007) CCL parser, available at `http://www.seggu.net/ccl/`

[9] `http://news.google.com/`

5 Linguistic Analysis of Mark-Up

Is there a connection between mark-up and syntactic structure? Previous work (Barr et al., 2008) has only examined search engine queries, showing that they consist predominantly of short noun phrases. If web mark-up shared a similar characteristic, it might not provide sufficiently disambiguating cues to syntactic structure: HTML tags could be too short (e.g., singletons like "click <a>here") or otherwise unhelpful in resolving truly difficult ambiguities (such as PP-attachment). We began simply by counting various basic events in BLOG_p.

	Count	*POS Sequence*	*Frac*	*Sum*
1	1,242	NNP NNP	16.1%	
2	643	NNP	8.3	24.4
3	419	NNP NNP NNP	5.4	29.8
4	414	NN	5.4	35.2
5	201	JJ NN	2.6	37.8
6	138	DT NNP NNP	1.8	39.5
7	138	NNS	1.8	41.3
8	112	JJ	1.5	42.8
9	102	VBD	1.3	44.1
10	92	DT NNP NNP NNP	1.2	45.3
11	85	JJ NNS	1.1	46.4
12	79	NNP NN	1.0	47.4
13	76	NN NN	1.0	48.4
14	61	VBN	0.8	49.2
15	60	NNP NNP NNP NNP	0.8	50.0
BLOG_p	+3,869	*more with Count* ≤ 49	50.0%	

Table 3: Top 50% of marked POS tag sequences.

	Count	*Non-Terminal*	*Frac*	*Sum*
1	5,759	NP	74.5%	
2	997	VP	12.9	87.4
3	524	S	6.8	94.2
4	120	PP	1.6	95.7
5	72	ADJP	0.9	96.7
6	61	FRAG	0.8	97.4
7	41	ADVP	0.5	98.0
8	39	SBAR	0.5	98.5
9	19	PRN	0.2	98.7
10	18	NX	0.2	99.0
BLOG_p	+81	*more with Count* ≤ 16	1.0%	

Table 4: Top 99% of dominating non-terminals.

5.1 Surface Text Statistics

Out of 57,809 sentences, 6,047 (10.5%) are annotated (see Table 2); and 4,934 (8.5%) have multi-token bracketings. We do not distinguish HTML tags and track only unique bracketing end-points within a sentence. Of these, 6,015 are multi-token — an average per-sentence yield of 10.4%.[10]

[10] A non-trivial fraction of our corpus is older (pre-internet) unannotated articles, so this estimate may be conservative.

As expected, many of the annotated words are nouns, but there are adjectives, verbs and other parts of speech too (see Table 3). Mark-up is short, typically under five words, yet (by far) the most frequently marked sequence of POS tags is a pair.

5.2 Common Syntactic Subtrees

For three-quarters of all mark-up, the lowest dominating non-terminal is a noun phrase (see Table 4); there are also non-trace quantities of verb phrases (12.9%) and other phrases, clauses and fragments.

Of the top fifteen — *35.2%* of all — annotated productions, only one is *not* a noun phrase (see Table 5, left). Four of the fifteen lowest dominating non-terminals do *not* match the entire bracketing — all four miss the leading determiner, as we saw earlier. In such cases, we recursively split internal nodes until the bracketing aligned, as follows:

[S [NP the <a>*Toronto Star*][VP reports [NP this]
[PP in the softest possible way],[S stating ...]]]

S → NP VP → DT NNP NNP VBZ NP PP S

We can summarize productions more compactly by using a dependency framework and clipping off any dependents whose subtrees do not cross a bracketing boundary, relative to the parent. Thus,

DT NNP NNP VBZ DT IN DT JJS JJ NN

becomes DT NNP VBZ, "the <a>*Star* reports." Viewed this way, the top fifteen (now collapsed) productions cover *59.4%* of all cases and include four verb heads, in addition to a preposition and an adjective (see Table 5, right). This exposes five cases of inexact matches, three of which involve neglected determiners or adjectives to the left of the head. In fact, the only case that cannot be explained by dropped dependents is #8, where the daughters are marked but the parent is left out. Most instances contributing to this pattern are flat NPs that end with a noun, incorrectly assumed to be the head of *all* other words in the phrase, e.g.,

... [NP a 1994 *New Yorker* article] ...

As this example shows, disagreements (as well as agreements) between mark-up and machine-generated parse trees with automatically percolated heads should be taken with a grain of salt.[11]

[11] In a relatively recent study, Ravi et al. (2008) report that Charniak's re-ranking parser (Charniak and Johnson, 2005) — `reranking-parserAug06.tar.gz`, also available from `ftp://ftp.cs.brown.edu/pub/nlparser/` — attains 86.3% accuracy when trained on WSJ and tested against Brown; its nearly 5% performance loss out-of-domain is consistent with the numbers originally reported by Gildea (2001).

	Count	Constituent Production	Frac	Sum
1	746	NP → NNP NNP	9.6%	
2	357	NP → NNP	4.6	14.3
3	266	NP → NP PP	3.4	17.7
4	183	NP → NNP NNP NNP	2.4	20.1
5	165	NP → DT NNP NNP	2.1	22.2
6	140	NP → NN	1.8	24.0
7	131	NP → DT NNP NNP NNP	1.7	25.7
8	130	NP → DT NN	1.7	27.4
9	127	NP → DT NNP NNP	1.6	29.0
10	109	S → NP VP	1.4	30.4
11	91	NP → DT NNP NNP NNP	1.2	31.6
12	82	NP → DT JJ NN	1.1	32.7
13	79	NP → NNS	1.0	33.7
14	65	NP → JJ NN	0.8	34.5
15	60	NP → NP NP	0.8	35.3
BLOG$_p$	+5,000	*more with Count* ≤ 60	64.7%	

	Count	Head-Outward Spawn	Frac	Sum
1	1,889	NNP	24.4%	
2	623	NN	8.1	32.5
3	470	DT NNP	6.1	38.6
4	458	DT NN	5.9	44.5
5	345	NNS	4.5	49.0
6	109	NNPS	1.4	50.4
7	98	VBG	1.3	51.6
8	96	NNP NNP NN	1.2	52.9
9	80	VBD	1.0	53.9
10	77	IN	1.0	54.9
11	74	VBN	1.0	55.9
12	73	DT JJ NN	0.9	56.8
13	71	VBZ	0.9	57.7
14	69	POS NNP	0.9	58.6
15	63	JJ	0.8	59.4
BLOG$_p$	+3,136	*more with Count* ≤ 62	40.6%	

Table 5: Top 15 marked productions, viewed as constituents (left) and as dependencies (right), after recursively expanding any internal nodes that did not align with the bracketing (underlined). Tabulated dependencies were collapsed, dropping any dependents that fell entirely in the same region as their parent (i.e., both inside the bracketing, both to its left or both to its right), keeping only crossing attachments.

5.3 Proposed Parsing Constraints

The straight-forward approach — forcing mark-up to correspond to constituents — agrees with Charniak's parse trees only **48.0**% of the time, e.g.,

... in [NP <a> [NP an analysis] [PP of perhaps the most astonishing PC item I have yet stumbled upon]].

This number should be higher, as the vast majority of disagreements are due to tree-bank idiosyncrasies (e.g., bare NPs). Earlier examples of incomplete constituents (e.g., legitimately missing determiners) would also be fine in many linguistic theories (e.g., as N-bars). A dependency formulation is less sensitive to such stylistic differences.

We begin with the hardest possible constraint on dependencies, then slowly relax it. Every example used to demonstrate a softer constraint doubles as a counter-example against all previous versions.

• *strict* — seals mark-up into attachments, i.e., inside a bracketing, enforces exactly one external arc — into the overall head. This agrees with head-percolated trees just **35.6**% of the time, e.g.,

As author of *The Satanic Verses*, I ...

• *loose* — same as *strict*, but allows the bracketing's head word to have external dependents. This relaxation already agrees with head-percolated dependencies **87.5**% of the time, catching many (though far from all) dropped dependents, e.g.,

... the *Toronto Star* reports ...

• *sprawl* — same as *loose*, but now allows *all* words inside a bracketing to attach external dependents.[12] This boosts agreement with head-percolated trees to **95.1**%, handling new cases, e.g., where "*Toronto Star*" is embedded in longer mark-up that includes its own parent — a verb:

... the <a>*Toronto Star* reports ... ...

• *tear* — allows mark-up to fracture after all, requiring only that the external heads attaching the pieces lie to the same side of the bracketing. This propels agreement with percolated dependencies to **98.9**%, fixing previously broken PP-attachment ambiguities, e.g., a fused phrase like "Fox News in Canada" that detached a preposition from its verb:

... concession ... has raised eyebrows among those waiting [PP for <a>Fox News] [PP in Canada] .

Most of the remaining 1.1% of disagreements are due to parser errors. Nevertheless, it *is* possible for mark-up to be torn apart by external heads from *both* sides. We leave this section with a (very rare) true negative example. Below, "CSA" modifies "authority" (to its left), appositively, while "Al-Manar" modifies "television" (to its right):[13]

The French broadcasting authority, <a>CSA, banned ... Al-Manar satellite television from ...

[12] This view evokes the trapezoids of the $O(n^3)$ recognizer for split head automaton grammars (Eisner and Satta, 1999).

[13] But this is a stretch, since the comma after "CSA" renders the marked phrase ungrammatical even *out* of context.

6 Experimental Methods and Metrics

We implemented the DMV (Klein and Manning, 2004), consulting the details of (Spitkovsky et al., 2010a). Crucially, we swapped out inside-outside re-estimation in favor of Viterbi training. Not only is it better-suited to the general problem (see §7.1), but it also admits a trivial implementation of (most of) the dependency constraints we proposed.[14]

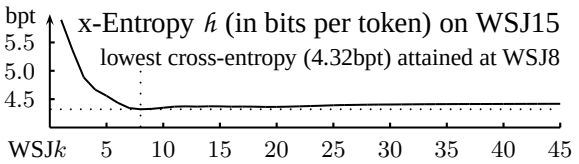


Figure 1: Sentence-level cross-entropy on WSJ15 for Ad-Hoc* initializers of WSJ$\{1, \ldots, 45\}$.

Six settings parameterized each run:

• INIT: 0 — default, uniform initialization; or 1 — a high quality initializer, pre-trained using Ad-Hoc* (Spitkovsky et al., 2010a): we chose the Laplace-smoothed model trained at WSJ15 (the "sweet spot" data gradation) but initialized off WSJ8, since that ad-hoc harmonic initializer has the best cross-entropy on WSJ15 (see Figure 1).

• GENRE: 0 — default, baseline training on WSJ; else, uses 1 — BLOG$_t$; 2 — NEWS; or 3 — WEB.

• SCOPE: 0 — default, uses all sentences up to length 45; if 1, trains using sentences up to length 15; if 2, re-trains on sentences up to length 45, starting from the solution to sentences up to length 15, as recommended by Spitkovsky et al. (2010a).

• CONSTR: if 4, *strict*; if 3, *loose*; and if 2, *sprawl*. We did not implement level 1, *tear*. Over-constrained sentences are re-attempted at successively lower levels until they become possible to parse, if necessary at the lowest (default) level 0.[15]

• TRIM: if 1, discards any sentence without a single multi-token mark-up (shorter than its length).

• ADAPT: if 1, upon convergence, initializes re-training on WSJ45 using the solution to <GENRE>, attempting domain adaptation (Lee et al., 1991).

These make for 294 meaningful combinations. We judged each one by its accuracy on WSJ45, using standard directed scoring — the fraction of correct dependencies over randomized "best" parse trees.

[14] We analyze the benefits of Viterbi training in a companion paper (Spitkovsky et al., 2010b), which dedicates more space to implementation and to the WSJ baselines used here.

[15] At level 4, X<u> Y Z</u> is over-constrained.

7 Discussion of Experimental Results

Evaluation on Section 23 of WSJ and Brown reveals that blog-training beats all published state-of-the-art numbers in every traditionally-reported length cutoff category, with news-training not far behind. Here is a mini-preview of these results, for Section 23 of WSJ10 and WSJ$^\infty$ (from Table 8):

	WSJ10	WSJ$^\infty$
(Cohen and Smith, 2009)	62.0	42.2
(Spitkovsky et al., 2010a)	57.1	45.0
NEWS-best	67.3	*50.1*
BLOG$_t$-best	**69.3**	**50.4**
(Headden et al., 2009)	*68.8*	

Table 6: Directed accuracies on Section 23 of WSJ$\{10, \infty\}$ for three recent state-of-the-art systems and our best runs (as judged against WSJ45) for NEWS and BLOG$_t$ (more details in Table 8).

Since our experimental setup involved testing nearly three hundred models simultaneously, we must take extreme care in analyzing and interpreting these results, to avoid falling prey to any looming "data-snooping" biases.[16] In a sufficiently large pool of models, where each is trained using a randomized and/or chaotic procedure (such as ours), the best may look good due to pure chance. We appealed to three separate diagnostics to convince ourselves that our best results are *not* noise.

The most radical approach would be to write off WSJ as a development set and to focus only on the results from the held-out Brown corpus. It was initially intended as a test of out-of-domain generalization, but since Brown was in no way involved in selecting the best models, it also qualifies as a blind evaluation set. We observe that our best models perform even better (and gain more — see Table 8) on Brown than on WSJ — a strong indication that our selection process has not overfitted.

Our second diagnostic is a closer look at WSJ. Since we cannot graph the full (six-dimensional) set of results, we begin with a simple linear regression, using accuracy on WSJ45 as the dependent variable. We prefer this full factorial design to the more traditional ablation studies because it allows us to account for and to incorporate every single experimental data point incurred along the

[16] In the standard statistical hypothesis testing setting, it is reasonable to expect that $p\%$ of randomly chosen hypotheses will appear significant at the $p\%$ level simply by chance. Consequently, *multiple* hypothesis testing requires re-evaluating significance levels — adjusting raw p-values, e.g., using the Holm-Bonferroni method (Holm, 1979).

Corpus	*Marked Sentences*	*All Sentences*	*POS Tokens*	*All Bracketings*	*Multi-Token Bracketings*
BLOG$_t$45	5,641	56,191	1,048,404	7,021	5,346
BLOG$'_t$45	4,516	4,516	104,267	5,771	5,346
BLOG$_t$15	1,562	23,214	212,872	1,714	1,240
BLOG$'_t$15	1,171	1,171	11,954	1,288	1,240
NEWS45	304,129,910	2,263,563,078	32,119,123,561	611,644,606	477,362,150
NEWS$'$45	205,671,761	205,671,761	2,740,258,972	453,781,081	392,600,070
NEWS15	211,659,549	1,433,779,438	11,786,164,503	365,145,549	274,791,675
NEWS$'$15	147,848,358	147,848,358	1,397,562,474	272,223,918	231,029,921
WEB45	1,577,208,680	8,903,458,234	87,269,385,640	3,309,897,461	2,459,337,571
WEB$'$45	933,115,032	933,115,032	11,552,983,379	2,084,359,555	1,793,238,913
WEB15	1,181,696,194	7,488,669,239	55,014,582,024	2,071,743,595	1,494,675,520
WEB$'$15	681,087,020	681,087,020	5,813,555,341	1,200,980,738	1,072,910,682

Table 7: Counts of sentences, tokens and (unique) bracketings for web-based data sets; trimmed versions, restricted to only those sentences having at least one multi-token bracketing, are indicated by a prime ($'$).

way. Its output is a coarse, high-level summary of our runs, showing which factors significantly contribute to changes in error rate on WSJ45:

Parameter	*(Indicator)*	*Setting*	$\hat{\beta}$	*p-value*
INIT	1	ad-hoc @WSJ8,15	11.8	***
GENRE	1	BLOG$_t$	-3.7	0.06
	2	NEWS	-5.3	**
	3	WEB	-7.7	***
SCOPE	1	@15	-0.5	0.40
	2	@15→45	-0.4	0.53
CONSTR	2	*sprawl*	0.9	0.23
	3	*loose*	1.0	0.15
	4	*strict*	1.8	*
TRIM	1	drop unmarked	-7.4	***
ADAPT	1	WSJ re-training	1.5	**
Intercept		($R^2_{\text{Adjusted}} = 73.6\%$)	39.9	***

We use a standard convention: *** for $p < 0.001$;
** for $p < 0.01$ (very signif.); and * for $p < 0.05$ (signif.).

The default training mode (all parameters zero) is estimated to score 39.9%. A good initializer gives the biggest (double-digit) gain; both domain adaptation and constraints also make a positive impact. Throwing away unannotated data hurts, as does training out-of-domain (the blog is least bad; the web is worst). Of course, this overview should not be taken too seriously. Overly simplistic, a first order model ignores interactions between parameters. Furthermore, a least squares fit aims to capture central tendencies, whereas we are more interested in outliers — the best-performing runs.

A major imperfection of the simple regression model is that helpful factors that require an interaction to "kick in" may not, on their own, appear statistically significant. Our third diagnostic is to examine parameter settings that give rise to the best-performing models, looking out for combinations that consistently deliver superior results.

7.1 WSJ Baselines

Just two parameters apply to learning from WSJ. Five of their six combinations are state-of-the-art, demonstrating the power of Viterbi training; only the default run scores worse than 45.0%, attained by Leapfrog (Spitkovsky et al., 2010a), on WSJ45:

Settings	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	*41.3*	45.0	45.2
1	46.6	47.5	**47.6**
	@45	@15	@15→45

7.2 Blog

Simply training on BLOG$_t$ instead of WSJ hurts:

GENRE=1	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	39.6	36.9	36.9
1	46.5	46.3	46.4
	@45	@15	@15→45

The best runs use a good initializer, discard unannotated sentences, enforce the *loose* constraint on the rest, follow up with domain adaptation and benefit from re-training — GENRE=TRIM=ADAPT=1:

INIT=1	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	45.8	48.3	49.6
(*sprawl*) 2	46.3	49.2	49.2
(*loose*) 3	41.3	*50.2*	**50.4**
(*strict*) 4	40.7	49.9	48.7
	@45	@15	@15→45

The contrast between unconstrained learning and annotation-guided parsing is higher for the default initializer, still using trimmed data sets (just over a thousand sentences for BLOG$'_t$15 — see Table 7):

INIT=0	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	25.6	19.4	19.3
(*sprawl*) 2	25.2	22.7	22.5
(*loose*) 3	32.4	26.3	27.3
(*strict*) 4	36.2	38.7	40.1
	@45	@15	@15→45

Above, we see a clearer benefit to our constraints.

7.3 News

Training on WSJ is also better than using NEWS:

GENRE=2	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	40.2	38.8	38.7
1	43.4	44.0	43.8
	@45	@15	@15→45

As with the blog, the best runs use the good initializer, discard unannotated sentences, enforce the *loose* constraint and follow up with domain adaptation — GENRE=2; INIT=TRIM=ADAPT=1:

Settings	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	46.6	45.4	45.2
(*sprawl*) 2	46.1	44.9	44.9
(*loose*) 3	**49.5**	48.1	48.3
(*strict*) 4	37.7	36.8	37.6
	@45	@15	@15→45

With all the extra training data, the best new score is just 49.5%. On the one hand, we are disappointed by the lack of dividends to orders of magnitude more data. On the other, we are comforted that the system arrives within 1% of its best result — 50.4%, obtained with a manually cleaned up corpus — now using an auto-generated data set.

7.4 Web

The WEB-side story is more discouraging:

GENRE=3	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	38.3	35.1	35.2
1	42.8	43.6	43.4
	@45	@15	@15→45

Our best run again uses a good initializer, keeps *all* sentences, still enforces the *loose* constraint and follows up with domain adaptation, but performs worse than all well-initialized WSJ baselines, scoring only 45.9% (trained at WEB15).

We suspect that the web is just too messy for us. On top of the challenges of language identification and sentence-breaking, there is a lot of boiler-plate; furthermore, web text can be difficult for news-trained POS taggers. For example, note that the verb "sign" is twice mistagged as a noun and that "YouTube" is classified as a verb, in the top four POS sequences of web sentences:[17]

	POS Sequence	WEB *Count*
	Sample web sentence, chosen uniformly at random.	
1	DT NNS VBN	82,858,487
	All rights reserved.	
2	NNP NNP NNP	65,889,181
	Yuasa et al.	
3	NN IN TO VB RB	31,007,783
	Sign in to YouTube now!	
4	NN IN IN PRP$ JJ NN	31,007,471
	Sign in with your Google Account!	

[17] Further evidence: TnT tags the ubiquitous but ambiguous fragments "click here" and "print post" as noun phrases.

7.5 The State of the Art

Our best model gains more than 5% over previous state-of-the-art accuracy across all sentences of WSJ's Section 23, more than 8% on WSJ20 and rivals the oracle skyline (Spitkovsky et al., 2010a) on WSJ10; these gains generalize to Brown100, where it improves by nearly 10% (see Table 8).

We take solace in the fact that our best models agree in using *loose* constraints. Of these, the models trained with less data perform better, with the best two using trimmed data sets, echoing that "less is more" (Spitkovsky et al., 2010a), pace Halevy et al. (2009). We note that orders of magnitude more data did not improve parsing performance further and suspect a different outcome from lexicalized models: The primary benefit of additional lower-quality data is in improved coverage. But with only 35 unique POS tags, data sparsity is hardly an issue. Extra examples of lexical items help little and hurt when they are mistagged.

8 Related Work

The wealth of new annotations produced in many languages every day already fuels a number of NLP applications. Following their early and wide-spread use by search engines, in service of spam-fighting and retrieval, anchor text and link data enhanced a variety of traditional NLP techniques: cross-lingual information retrieval (Nie and Chen, 2002), translation (Lu et al., 2004), both named-entity recognition (Mihalcea and Csomai, 2007) and categorization (Watanabe et al., 2007), query segmentation (Tan and Peng, 2008), plus semantic relatedness and word-sense disambiguation (Gabrilovich and Markovitch, 2007; Yeh et al., 2009). Yet several, seemingly natural, candidate core NLP tasks — tokenization, CJK segmentation, noun-phrase chunking, and (until now) parsing — remained conspicuously uninvolved.

Approaches related to ours arise in applications that combine parsing with named-entity recognition (NER). For example, constraining a parser to respect the boundaries of known entities is standard practice not only in joint modeling of (constituent) parsing and NER (Finkel and Manning, 2009), but also in higher-level NLP tasks, such as relation extraction (Mintz et al., 2009), that couple chunking with (dependency) parsing. Although restricted to proper noun phrases, dates, times and quantities, we suspect that constituents identified by trained (supervised) NER systems would also

Model	*Incarnation*		WSJ10	WSJ20	WSJ$^{\infty}$	
DMV	Bilingual Log-Normals (tie-verb-noun)	(Cohen and Smith, 2009)	62.0	48.0	42.2	Brown100
	Leapfrog	(Spitkovsky et al., 2010a)	57.1	48.7	45.0	43.6
	default	INIT=0,GENRE=0,SCOPE=0,CONSTR=0,TRIM=0,ADAPT=0	55.9	45.8	41.6	40.5
	WSJ-best	INIT=1,GENRE=0,SCOPE=2,CONSTR=0,TRIM=0,ADAPT=0	65.3	53.8	47.9	50.8
	BLOG$_t$-best	INIT=1,GENRE=1,SCOPE=2,CONSTR=3,TRIM=1,ADAPT=1	**69.3**	**56.8**	**50.4**	**53.3**
	NEWS-best	INIT=1,GENRE=2,SCOPE=0,CONSTR=3,TRIM=1,ADAPT=1	67.3	*56.2*	*50.1*	*51.6*
	WEB-best	INIT=1,GENRE=3,SCOPE=1,CONSTR=3,TRIM=0,ADAPT=1	64.1	52.7	46.3	46.9
EVG	Smoothed (skip-head), Lexicalized	(Headden et al., 2009)	*68.8*			

Table 8: Accuracies on Section 23 of WSJ$\{10, 20, \infty\}$ and Brown100 for three recent state-of-the-art systems, our default run, and our best runs (judged by accuracy on WSJ45) for each of four training sets.

be helpful in constraining grammar induction.

Following Pereira and Schabes' (1992) success with partial annotations in training a model of (English) constituents generatively, their idea has been extended to discriminative estimation (Riezler et al., 2002) and also proved useful in modeling (Japanese) dependencies (Sassano, 2005). There was demand for partially bracketed corpora. Chen and Lee (1995) constructed one such corpus by learning to partition (English) POS sequences into chunks (Abney, 1991); Inui and Kotani (2001) used n-gram statistics to split (Japanese) clauses.

We combine the two intuitions, using the web to build a partially parsed corpus. Our approach could be called *lightly-supervised*, since it does not require manual annotation of a single complete parse tree. In contrast, traditional semi-supervised methods rely on fully-annotated seed corpora.[18]

9 Conclusion

We explored novel ways of training dependency parsing models, the best of which attains 50.4% accuracy on Section 23 (all sentences) of WSJ, beating all previous unsupervised state-of-the-art by more than 5%. Extra gains stem from guiding Viterbi training with web mark-up, the *loose* constraint consistently delivering best results. Our linguistic analysis of a blog reveals that web annotations can be converted into accurate parsing constraints (*loose*: 88%; *sprawl*: 95%; *tear*: 99%) that could be helpful to supervised methods, e.g., by boosting an initial parser via self-training (McClosky et al., 2006) on sentences with mark-up. Similar techniques may apply to standard word-processing annotations, such as font changes, and to certain (balanced) punctuation (Briscoe, 1994).

We make our blog data set, overlaying mark-up and syntax, publicly available. Its annotations are 75% noun phrases, 13% verb phrases, 7% simple declarative clauses and 2% prepositional phrases, with traces of other phrases, clauses and fragments. The type of mark-up, combined with POS tags, could make for valuable features in discriminative models of parsing (Ratnaparkhi, 1999).

A logical next step would be to explore the connection between syntax and mark-up for genres other than a news-style blog and for languages other than English. We are excited by the possibilities, as unsupervised parsers are on the cusp of becoming useful in their own right — recently, Davidov et al. (2009) successfully applied Seginer's (2007) fully unsupervised grammar inducer to the problems of pattern-acquisition and extraction of semantic data. If the strength of the connection between web mark-up and syntactic structure is universal across languages and genres, this fact could have broad implications for NLP, with applications extending well beyond parsing.

[18] A significant effort expended in building a tree-bank comes with the first batch of sentences (Druck et al., 2009).

Acknowledgments

Partially funded by NSF award IIS-0811974 and by the Air Force Research Laboratory (AFRL), under prime contract no. FA8750-09-C-0181; first author supported by the Fannie & John Hertz Foundation Fellowship. We thank Angel X. Chang, Spence Green, Christopher D. Manning, Richard Socher, Mihai Surdeanu and the anonymous reviewers for many helpful suggestions, and we are especially grateful to Andy Golding, for pointing us to his sample Map-Reduce over the Google News crawl, and to Daniel Pipes, for allowing us to distribute the data set derived from his blog entries.

References

S. Abney. 1991. Parsing by chunks. *Principle-Based Parsing: Computation and Psycholinguistics*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*.

C. Barr, R. Jones, and M. Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*.

T. Brants. 2000. TnT — a statistical part-of-speech tagger. In *ANLP*.

T. Briscoe. 1994. Parsing (with) punctuation, etc. Technical report, Xerox European Research Laboratory.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*.

E. Charniak. 2001. Immediate-head parsing for language models. In *ACL*.

H.-H. Chen and Y.-S. Lee. 1995. Development of a partially bracketed corpus with part-of-speech information only. In *WVLC*.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL-HLT*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

D. Davidov, R. Reichart, and A. Rappoport. 2009. Superior and efficient fully unsupervised pattern-based concept acquisition using an unsupervised parser. In *CoNLL*.

G. Druck, G. Mann, and A. McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *ACL-IJCNLP*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *ACL*.

J. R. Finkel and C. D. Manning. 2009. Joint parsing and named entity recognition. In *NAACL-HLT*.

W. N. Francis and H. Kucera, 1979. *Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistic, Brown University.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*.

D. Gildea. 2001. Corpus variation and parser performance. In *EMNLP*.

A. Halevy, P. Norvig, and F. Pereira. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

S. Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6.

N. Inui and Y. Kotani. 2001. Robust N-gram based syntactic analysis using segmentation words. In *PACLIC*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

C.-H. Lee, C.-H. Lin, and B.-H. Juang. 1991. A study on speaker adaptation of the parameters of continuous density Hidden Markov Models. *IEEE Trans. on Signal Processing*, 39.

W.-H. Lu, L.-F. Chien, and H.-J. Lee. 2004. Anchor text mining for translation of Web queries: A transitive translation approach. *ACM Trans. on Information Systems*, 22.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

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

R. Mihalcea and A. Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *CIKM*.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*.

J.-Y. Nie and J. Chen. 2002. Exploiting the Web as parallel corpora for cross-language information retrieval. *Web Intelligence*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34.

S. Ravi, K. Knight, and R. Soricut. 2008. Automatic prediction of parser accuracy. In *EMNLP*.

J. C. Reynar and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *ANLP*.

S. Riezler, T. H. King, R. M. Kaplan, R. Crouch, J. T. Maxwell, III, and M. Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *ACL*.

M. Sassano. 2005. Using a partially annotated corpus to build a dependency parser for Japanese. In *IJCNLP*.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2010a. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *NAACL-HLT*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

B. Tan and F. Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW*.

K. Toutanova and C. D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP-VLC*.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

D. Vadas and J. R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *ACL*.

Y. Watanabe, M. Asahara, and Y. Matsumoto. 2007. A graph-based approach to named entity categorization in Wikipedia using conditional random fields. In *EMNLP-CoNLL*.

E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa. 2009. WikiWalk: Random walks on Wikipedia for semantic relatedness. In *TextGraphs*.

Phylogenetic Grammar Induction

Taylor Berg-Kirkpatrick and **Dan Klein**
Computer Science Division
University of California, Berkeley
{tberg, klein}@cs.berkeley.edu

Abstract

We present an approach to multilingual grammar induction that exploits a phylogeny-structured model of parameter drift. Our method does not require any translated texts or token-level alignments. Instead, the phylogenetic prior couples languages at a parameter level. Joint induction in the multilingual model substantially outperforms independent learning, with larger gains both from more articulated phylogenies and as well as from increasing numbers of languages. Across eight languages, the multilingual approach gives error reductions over the standard monolingual DMV averaging 21.1% and reaching as high as 39%.

1 Introduction

Learning multiple languages together should be easier than learning them separately. For example, in the domain of syntactic parsing, a range of recent work has exploited the mutual constraint between two languages' parses of the same bitext (Kuhn, 2004; Burkett and Klein, 2008; Kuzman et al., 2009; Smith and Eisner, 2009; Snyder et al., 2009a). Moreover, Snyder et al. (2009b) in the context of unsupervised part-of-speech induction (and Bouchard-Côté et al. (2007) in the context of phonology) show that extending beyond two languages can provide increasing benefit. However, multitexts are only available for limited languages and domains. In this work, we consider unsupervised grammar induction without bitexts or multitexts. Without translation examples, multilingual constraints cannot be exploited at the sentence token level. Rather, we capture multilingual constraints at a *parameter* level, using a phylogeny-structured prior to tie together the various individual languages' learning problems. Our joint, hierarchical prior couples model parameters for different languages in a way that respects knowledge about how the languages evolved.

Aspects of this work are closely related to Cohen and Smith (2009) and Bouchard-Côté et al. (2007). Cohen and Smith (2009) present a model for jointly learning English and Chinese dependency grammars without bitexts. In their work, structurally constrained covariance in a logistic normal prior is used to couple parameters between the two languages. Our work, though also different in technical approach, differs most centrally in the extension to multiple languages and the use of a phylogeny. Bouchard-Côté et al. (2007) considers an entirely different problem, phonological reconstruction, but shares with this work both the use of a phylogenetic structure as well as the use of log-linear parameterization of local model components. Our work differs from theirs primarily in the task (syntax vs. phonology) and the variables governed by the phylogeny: in our model it is the grammar parameters that drift (in the prior) rather than individual word forms (in the likelihood model).

Specifically, we consider dependency induction in the DMV model of Klein and Manning (2004). Our data is a collection of standard dependency data sets in eight languages: English, Dutch, Danish, Swedish, Spanish, Portuguese, Slovene, and Chinese. Our focus is not the DMV model itself, which is well-studied, but rather the prior which couples the various languages' parameters. While some choices of prior structure can greatly complicate inference (Cohen and Smith, 2009), we choose a hierarchical Gaussian form for the drift term, which allows the gradient of the observed data likelihood to be easily computed using standard dynamic programming methods.

In our experiments, joint multilingual learning substantially outperforms independent monolingual learning. Using a limited phylogeny that

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1288–1297,
Uppsala, Sweden, 11-16 July 2010.

only couples languages within linguistic families reduces error by 5.6% over the monolingual baseline. Using a flat, global phylogeny gives a greater reduction, almost 10%. Finally, a more articulated phylogeny that captures both inter- and intra-family effects gives an even larger average relative error reduction of 21.1%.

2 Model

We define our model over two kinds of random variables: dependency trees and parameters. For each language ℓ in a set L, our model will generate a collection $\mathbf{t}_\ell$ of dependency trees $\mathbf{t}_\ell^i$. We assume that these dependency trees are generated by the DMV model of Klein and Manning (2004), which we write as $\mathbf{t}_\ell^i \sim \text{DMV}(\theta_\ell)$. Here, θ_ℓ is a vector of the various model parameters for language ℓ. The prior is what couples the θ_ℓ parameter vectors across languages; it is the focus of this work. We first consider the likelihood model before moving on to the prior.

2.1 Dependency Model with Valence

A dependency parse is a directed tree $\mathbf{t}$ over tokens in a sentence $\mathbf{s}$. Each edge of the tree specifies a directed dependency from a head token to a dependent, or argument token. The DMV is a generative model for trees $\mathbf{t}$, which has been widely used for dependency parse induction. The observed data likelihood, used for parameter estimation, is the marginal probability of generating the observed sentences $\mathbf{s}$, which are simply the leaves of the trees $\mathbf{t}$. Generation in the DMV model involves two types of local conditional probabilities: CONTINUE distributions that capture valence and ATTACH distributions that capture argument selection.

First, the Bernoulli CONTINUE probability distributions $P^{\text{CONTINUE}}(c|h, dir, adj; \theta_\ell)$ model the fertility of a particular head type h. The outcome $c \in \{stop, continue\}$ is conditioned on the head type h, direction dir, and adjacency adj. If a head type's continue probability is low, tokens of this type will tend to generate few arguments.

Second, the ATTACH multinomial probability distributions $P^{\text{ATTACH}}(a|h, dir; \theta_\ell)$ capture attachment preferences of heads, where a and h are both token types. We take the same approach as previous work (Klein and Manning, 2004; Cohen and Smith, 2009) and use gold part-of-speech labels as tokens. Thus, the basic observed "word" types are actually word classes.

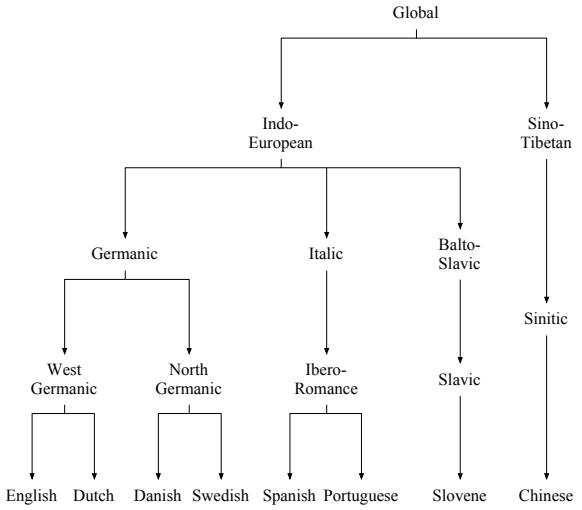


Figure 1: An example of a linguistically-plausible phylogenetic tree over the languages in our training data. Leaves correspond to (observed) modern languages, while internal nodes represent (unobserved) ancestral languages.

2.1.1 Log-Linear Parameterization

The DMV's local conditional distributions were originally given as simple multinomial distributions with one parameter per outcome. However, they can be re-parameterized to give the following log-linear form (Eisner, 2002; Bouchard-Côté et al., 2007; Berg-Kirkpatrick et al., 2010):

$$P^{\text{CONTINUE}}(c|h, dir, adj; \theta_\ell) = \frac{\exp\left[{\theta_\ell}^T \boldsymbol{f}_{\text{CONTINUE}}(c, h, dir, adj)\right]}{\sum_{c'} \exp\left[{\theta_\ell}^T \boldsymbol{f}_{\text{CONTINUE}}(c', h, dir, adj)\right]}$$

$$P^{\text{ATTACH}}(a|h, dir; \theta_\ell) = \frac{\exp\left[{\theta_\ell}^T \boldsymbol{f}_{\text{ATTACH}}(a, h, dir)\right]}{\sum_{a'} \exp\left[{\theta_\ell}^T \boldsymbol{f}_{\text{ATTACH}}(a', h, dir)\right]}$$

The parameters are weights θ_ℓ with one weight vector per language. In the case where the vector of feature functions $\boldsymbol{f}$ has an indicator for each possible conjunction of outcome and conditions, the original multinomial distributions are recovered. We refer to these full indicator features as the set of SPECIFIC features.

2.2 Phylogenetic Prior

The focus of this work is coupling each of the parameters θ_ℓ in a phylogeny-structured prior. Consider a phylogeny like the one shown in Figure 1, where each modern language ℓ in L is a leaf. We would like to say that the leaves' parameter vectors arise from a process which slowly

drifts along each branch. A convenient choice is to posit additional parameter variables θ_{ℓ^+} at internal nodes $\ell^+ \in L^+$, a set of ancestral languages, and to assume that the conditional distribution $P(\theta_\ell|\theta_{\text{par}(\ell)})$ at each branch in the phylogeny is a Gaussian centered on $\theta_{\text{par}(\ell)}$, where $\text{par}(\ell)$ is the parent of ℓ in the phylogeny and ℓ ranges over $L \cup L^+$. The variance structure of the Gaussian would then determine how much drift (and in what directions) is expected. Concretely, we assume that each drift distribution is an isotropic Gaussian with mean $\theta_{\text{par}(\ell)}$ and scalar variance σ^2. The root is centered at zero. We have thus defined a joint distribution $P(\Theta|\sigma^2)$ where $\Theta = (\theta_\ell : \ell \in L \cup L^+)$. σ^2 is a hyperparameter for this prior which could itself be re-parameterized to depend on branch length or be learned; we simply set it to a plausible constant value.

Two primary challenges remain. First, inference under arbitrary priors can become complex. However, in the simple case of our diagonal covariance Gaussians, the gradient of the observed data likelihood can be computed directly using the DMV's expected counts and maximum-likelihood estimation can be accomplished by applying standard gradient optimization methods. Second, while the choice of diagonal covariance is efficient, it causes components of θ that correspond to features occurring in only one language to be marginally independent of the parameters of all other languages. In other words, only features which fire in more than one language are coupled by the prior. In the next section, we therefore increase the overlap between languages' features by using coarse projections of parts-of-speech.

2.3 Projected Features

With diagonal covariance in the Gaussian drift terms, each parameter evolves independently of the others. Therefore, our prior will be most informative when features activate in multiple languages. In phonology, it is useful to map phonemes to the International Phonetic Alphabet (IPA) in order to have a language-independent parameterization. We introduce a similarly neutral representation here by projecting language-specific parts-of-speech to a coarse, shared inventory.

Indeed, we assume that each language has a distinct tagset, and so the basic configurational features will be language specific. For example, when an English VBZ takes a left argument headed by a NNS, a feature will activate specific to VBZ-NNS-LEFT. That feature will be used in the log-linear attachment probability for English. However, because that feature does not show up in any other language, it is not usefully controlled by the prior. Therefore, we also include coarser features which activate on more abstract, cross-linguistic configurations. In the same example, a feature will fire indicating a coarse, direction-free NOUN-VERB attachment. This feature will now occur in multiple languages and will contribute to each of those languages' attachment models. Although such cross-lingual features will have different weight parameters in each language, those weights will covary, being correlated by the prior.

SPECIFIC:	Activate for only one conjunction of outcome and conditions: $\mathbb{1}(c = \cdot, h = \cdot, dir = \cdot, adj = \cdot)$
SHARED:	Activate for heads from multiple languages using cross-lingual POS projection $\pi(\cdot)$: $\mathbb{1}(c = \cdot, \pi(h) = \cdot, dir = \cdot, adj = \cdot)$

CONTINUE distribution feature templates.

SPECIFIC:	Activate for only one conjunction of outcome and conditions: $\mathbb{1}(a = \cdot, h = \cdot, dir = \cdot)$
SHARED:	Activate for heads and arguments from multiple languages using cross-lingual POS projection $\pi(\cdot)$: $\mathbb{1}(\pi(a) = \cdot, \pi(h) = \cdot, dir = \cdot)$ $\mathbb{1}(\pi(a) = \cdot, h = \cdot, dir = \cdot)$ $\mathbb{1}(a = \cdot, \pi(h) = \cdot, dir = \cdot)$

ATTACH distribution feature templates.

Table 1: Feature templates for CONTINUE and ATTACH conditional distributions.

The coarse features are defined via a projection π from language-specific part-of-speech labels to coarser, cross-lingual word classes, and hence we refer to them as SHARED features. For each corpus used in this paper, we use the tagging annotation guidelines to manually define a fixed mapping from the corpus tagset to the following coarse tagset: noun, verb, adjective, adverb, conjunction, preposition, determiner, interjection, numeral, and pronoun. Parts-of-speech for which this coarse mapping is ambiguous or impossible are not mapped, and do not have corresponding SHARED features.

We summarize the feature templates for the CONTINUE and ATTACH conditional distributions in Table 1. Variants of all feature templates that ignore direction and/or adjacency are included. In practice, we found it beneficial for all language-

independent features to ignore direction.

Again, only the coarse features occur in multiple languages, so all phylogenetic influence is through those. Nonetheless, the effect of the phylogeny turns out to be quite strong.

2.4 Learning

We now turn to learning with the phylogenetic prior. Since the prior couples parameters across languages, this learning problem requires parameters for all languages be estimated jointly. We seek to find $\Theta = (\theta_\ell : \ell \in L \cup L^+)$ which optimizes $\log P(\Theta|\mathbf{s})$, where $\mathbf{s}$ aggregates the observed leaves of all the dependency trees in all the languages. This can be written as

$$\log P(\Theta) + \log P(\mathbf{s}|\Theta) - \log P(\mathbf{s})$$

The third term is a constant and can be ignored. The first term can be written as

$$\log P(\Theta) = \sum_{\ell \in L \cup L^+} \frac{1}{2\sigma^2} \|\theta_\ell - \theta_{\mathrm{par}(\ell)}\|_2^2 + C$$

where C is a constant. The form of $\log P(\Theta)$ immediately shows how parameters are penalized for being different across languages, more so for languages that are near each other in the phylogeny. The second term

$$\log P(\mathbf{s}|\Theta) = \sum_{\ell \in L} \log P(\mathbf{s}_\ell|\theta_\ell)$$

is a sum of observed data likelihoods under the standard DMV models for each language, computable by dynamic programming (Klein and Manning, 2004). Together, this yields the following objective function:

$$l(\Theta) = \textstyle\sum_{\ell \in L \cup L^+} \frac{1}{2\sigma^2} \|\theta_\ell - \theta_{\mathrm{par}(\ell)}\|_2^2 + \sum_{\ell \in L} \log P(\mathbf{s}_\ell|\theta_\ell)$$

which can be optimized using gradient methods or (MAP) EM. Here we used L-BFGS (Liu et al., 1989). This requires computation of the gradient of the observed data likelihood $\log P(\mathbf{s}_\ell|\theta_\ell)$ which is given by:

$$\nabla \log P(\mathbf{s}_\ell|\theta_\ell) = \mathbb{E}_{\mathbf{t}_\ell|\mathbf{s}_\ell}\left[\nabla \log P(\mathbf{s}_\ell, \mathbf{t}_\ell|\theta_\ell)\right] =$$

$$\begin{bmatrix} \sum_{c,h,dir,adj} e_{c,h,dir,adj}(\mathbf{s}_\ell;\theta_\ell) \cdot \Big[\boldsymbol{f}_{\text{CONTINUE}}(c,h,dir,adj) - \\ \quad \sum_{c'} P^{\text{CONTINUE}}(c'|h,dir,adj;\theta_\ell)\boldsymbol{f}_{\text{CONTINUE}}(c',h,dir,adj)\Big] \\ \\ \sum_{a,h,dir} e_{a,h,dir}(\mathbf{s}_\ell;\theta_\ell) \cdot \Big[\boldsymbol{f}_{\text{ATTACH}}(a,h,dir) - \\ \quad \sum_{a'} P^{\text{ATTACH}}(a'|h,dir;\theta_\ell)\boldsymbol{f}_{\text{ATTACH}}(a',h,dir)\Big] \end{bmatrix}$$

The expected gradient of the log joint likelihood of sentences and parses is equal to the gradient of the log marginal likelihood of just sentences, or the observed data likelihood (Salakhutdinov et al., 2003). $e_{a,h,dir}(\mathbf{s}_\ell;\theta_\ell)$ is the expected count of the number of times head h is attached to a in direction dir given the observed sentences $\mathbf{s}_\ell$ and DMV parameters θ_ℓ. $e_{c,h,dir,adj}(\mathbf{s}_\ell;\theta_\ell)$ is defined similarly. Note that these are the same expected counts required to perform EM on the DMV, and are computable by dynamic programming.

The computation time is dominated by the computation of each sentence's posterior expected counts, which are independent given the parameters, so the time required per iteration is essentially the same whether training all languages jointly or independently. In practice, the total number of iterations was also similar.

3 Experimental Setup

3.1 Data

We ran experiments with the following languages: English, Dutch, Danish, Swedish, Spanish, Portuguese, Slovene, and Chinese. For all languages but English and Chinese, we used corpora from the 2006 CoNLL-X Shared Task dependency parsing data set (Buchholz and Marsi, 2006). We used the shared task training set to both train and test our models. These corpora provide hand-labeled part-of-speech tags (except for Dutch, which is automatically tagged) and provide dependency parses, which are either themselves hand-labeled or have been converted from hand-labeled parses of other kinds. For English and Chinese we use sections 2-21 of the Penn Treebank (PTB) (Marcus et al., 1993) and sections 1-270 of the Chinese Treebank (CTB) (Xue et al., 2002) respectively. Similarly, these sections were used for both training and testing. The English and Chinese data sets have hand-labeled constituency parses and part-of-speech tags, but no dependency parses. We used the Bikel Chinese head finder (Bikel and Chiang, 2000) and the Collins English head finder (Collins, 1999) to transform the gold constituency parses into gold dependency parses. None of the corpora are bitexts. For all languages, we ran experiments on all sentences of length 10 or less after punctuation has been removed.

When constructing phylogenies over the languages we made use of their linguistic classifications. English and Dutch are part of the West Ger-

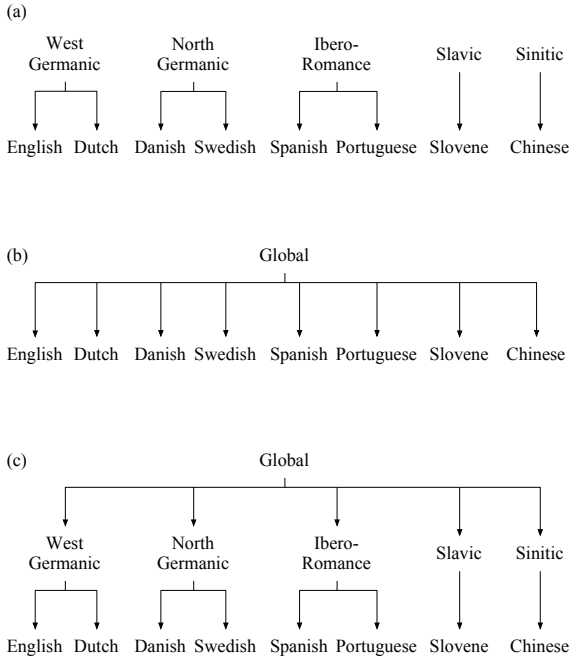


Figure 2: (a) Phylogeny for FAMILIES model. (b) Phylogeny for GLOBAL model. (c) Phylogeny for LINGUISTIC model.

manic family of languages, whereas Danish and Swedish are part of the North Germanic family. Spanish and Portuguese are both part of the Ibero-Romance family. Slovene is part of the Slavic family. Finally, Chinese is in the Sinitic family, and is not an Indo-European language like the others. We interchangeably speak of a language family and the ancestral node corresponding to that family's root language in a phylogeny.

3.2 Models Compared

We evaluated three phylogenetic priors, each with a different phylogenetic structure. We compare with two monolingual baselines, as well as an all-pairs multilingual model that does not have a phylogenetic interpretation, but which provides very similar capacity for parameter coupling.

3.2.1 Phylogenetic Models

The first phylogenetic model uses the shallow phylogeny shown in Figure 2(a), in which only languages within the same family have a shared parent node. We refer to this structure as FAMILIES. Under this prior, the learning task decouples into independent subtasks for each family, but no regularities across families can be captured.

The family-level model misses the constraints between distant languages. Figure 2(b) shows another simple configuration, wherein all languages share a common parent node in the prior, meaning that global regularities that are consistent across all languages can be captured. We refer to this structure as GLOBAL.

While the global model couples the parameters for all eight languages, it does so without sensitivity to the articulated structure of their descent. Figure 2(c) shows a more nuanced prior structure, LINGUISTIC, which groups languages first by family and then under a global node. This structure allows global regularities as well as regularities within families to be learned.

3.2.2 Parameterization and ALLPAIRS Model

Daumé III (2007) and Finkel and Manning (2009) consider a formally similar Gaussian hierarchy for domain adaptation. As pointed out in Finkel and Manning (2009), there is a simple equivalence between hierarchical regularization as described here and the addition of new tied features in a "flat" model with zero-meaned Gaussian regularization on all parameters. In particular, instead of parameterizing the objective in Section 2.4 in terms of multiple sets of weights, one at each node in the phylogeny (the *hierarchical parameterization*, described in Section 2.4), it is equivalent to parameterize this same objective in terms of a single set of weights on a larger of group features (the *flat parameterization*). This larger group of features contains a duplicate set of the features discussed in Section 2.3 for each node in the phylogeny, each of which is active only on the languages that are its descendants. A linear transformation between parameterizations gives equivalence. See Finkel and Manning (2009) for details.

In the flat parameterization, it seems equally reasonable to simply tie all pairs of languages by adding duplicate sets of features for each pair. This gives the ALLPAIRS setting, which we also compare to the tree-structured phylogenetic models above.

3.3 Baselines

To evaluate the impact of multilingual constraint, we compared against two monolingual baselines. The first baseline is the standard DMV with only SPECIFIC features, which yields the standard multinomial DMV (*weak baseline*). To facilitate comparison to past work, we used no prior for this monolingual model. The second baseline is the DMV with added SHARED features. This model includes a simple isotropic Gaussian prior on pa-

			Monolingual		Multilingual				
						Phylogenetic			
		Corpus Size	Baseline	Baseline w/ SHARED	ALLPAIRS	FAMILIES	BESTPAIR	GLOBAL	LINGUISTIC
West Germanic	English	6008	47.1	51.3	48.5	51.3	51.3 (Ch)	51.2	**62.3**
	Dutch	6678	36.3	36.0	44.0	36.1	36.2 (Sw)	44.0	**45.1**
North Germanic	Danish	1870	33.5	33.6	40.5	31.4	34.2 (Du)	39.6	**41.6**
	Swedish	3571	45.3	44.8	56.3	44.8	44.8 (Ch)	44.5	**58.3**
Ibero-Romance	Spanish	712	28.0	40.5	58.7	63.4	**63.8** (Da)	59.4	58.4
	Portuguese	2515	38.5	38.5	**63.1**	37.4	38.4 (Sw)	37.4	63.0
Slavic	Slovene	627	38.5	39.7	49.0	–	**49.6** (En)	49.4	48.4
Sinitic	Chinese	959	36.3	43.3	**50.7**	–	49.7 (Sw)	50.1	49.6
	Macro-Avg. Relative Error Reduction				17.1	5.6	8.5	9.9	**21.1**

Table 2: Directed dependency accuracy of monolingual and multilingual models, and relative error reduction over the monolingual baseline with SHARED features macro-averaged over languages. Multilingual models outperformed monolingual models in general, with larger gains from increasing numbers of languages. Additionally, more nuanced phylogenetic structures outperformed cruder ones.

rameters. This second baseline is the more direct comparison to the multilingual experiments here (*strong baseline*).

3.4 Evaluation

For each setting, we evaluated the directed dependency accuracy of the minimum Bayes risk (MBR) dependency parses produced by our models under maximum (posterior) likelihood parameter estimates. We computed accuracies separately for each language in each condition. In addition, for multilingual models, we computed the relative error reduction over the strong monolingual baseline, macro-averaged over languages.

3.5 Training

Our implementation used the flat parameterization described in Section 3.2.2 for both the phylogenetic and ALLPAIRS models. We originally did this in order to facilitate comparison with the non-phylogenetic ALLPAIRS model, which has no equivalent hierarchical parameterization. In practice, optimizing with the hierarchical parameterization also seemed to underperform.[1]

All models were trained by directly optimizing the observed data likelihood using L-BFGS (Liu et al., 1989). Berg-Kirkpatrick et al. (2010) suggest that directly optimizing the observed data likelihood may offer improvements over the more standard expectation-maximization (EM) optimization procedure for models such as the DMV, especially when the model is parameterized using features. We stopped training after 200 iterations in all cases. This fixed stopping criterion seemed to be adequate in all experiments, but presumably there is a potential gain to be had in fine tuning. To initialize, we used the harmonic initializer presented in Klein and Manning (2004). This type of initialization is deterministic, and thus we did not perform random restarts.

We found that for all models $\sigma^2 = 0.2$ gave reasonable results, and we used this setting in all experiments. For most models, we found that varying σ^2 in a reasonable range did not substantially affect accuracy. For some models, the directed accuracy was less flat with respect to σ^2. In these less-stable cases, there seemed to be an interaction between the variance and the choice between head conventions. For example, for some settings of σ^2, but not others, the model would learn that determiners head noun phrases. In particular, we observed that even when direct accuracy did fluctuate, undirected accuracy remained more stable.

[1] We noticed that the weights of features shared across languages had larger magnitude early in the optimization procedure when using the flat parameterization compared to using the hierarchical parameterization, perhaps indicating that cross-lingual influences had a larger effect on learning in its initial stages.

4 Results

Table 2 shows the overall results. In all cases, methods which coupled the languages in some way outperformed the independent baselines that considered each language independently.

4.1 Bilingual Models

The weakest of the coupled models was FAMILIES, which had an average relative error reduction of 5.6% over the strong baseline. In this case, most of the average improvement came from a single family: Spanish and Portuguese. The limited improvement of the family-level prior compared to other phylogenies suggests that there are important multilingual interactions that do not happen within families. Table 2 also reports the maximum accuracy achieved for each language when it was paired with another language (same family or otherwise) and trained together with a single common parent. These results appear in the column headed by BESTPAIR, and show the best accuracy for the language on that row over all possible pairings with other languages. When pairs of languages were trained together in isolation, the largest benefit was seen for languages with small training corpora, not necessarily languages with common ancestry. In our setup, Spanish, Slovene, and Chinese have substantially smaller training corpora than the rest of the languages considered. Otherwise, the patterns are not particularly clear; combined with subsequent results, it seems that pairwise constraint is fairly limited.

4.2 Multilingual Models

Models that coupled multiple languages performed better in general than models that only considered pairs of languages. The GLOBAL model, which couples all languages, if crudely, yielded an average relative error reduction of 9.9%. This improvement comes as the number of languages able to exert mutual constraint increases. For example, Dutch and Danish had large improvements, over and above any improvements these two languages gained when trained with a single additional language. Beyond the simplistic GLOBAL phylogeny, the more nuanced LINGUISTIC model gave large improvements for English, Swedish, and Portuguese. Indeed, the LINGUISTIC model is the only model we evaluated that gave improvements for *all* the languages we considered.

It is reasonable to worry that the improvements from these multilingual models might be partially due to having more total training data in the multilingual setting. However, we found that halving the amount of data used to train the English, Dutch, and Swedish (the languages with the most training data) monolingual models did not substantially affect their performance, suggesting that for languages with several thousand sentences or more, the increase in statistical support due to additional monolingual data was not an important effect (the DMV is a relatively low-capacity model in any case).

4.3 Comparison of Phylogenies

Recall the structures of the three phylogenies presented in Figure 2. These phylogenies differ in the correlations they can represent. The GLOBAL phylogeny captures only "universals," while FAMILIES captures only correlations between languages that are known to be similar. The LINGUISTIC model captures both of these effects simultaneously by using a two layer hierarchy. Notably, the improvement due to the LINGUISTIC model is more than the sum of the improvements due to the GLOBAL and FAMILIES models.

4.4 Phylogenetic vs. ALLPAIRS

The phylogeny is capable of allowing appropriate influence to pass between languages at multiple levels. We compare these results to the ALLPAIRS model in order to see whether limitation to a tree structure is helpful. The ALLPAIRS model achieved an average relative error reduction of 17.1%, certainly outperforming both the simple phylogenetic models. However, the rich phylogeny of the LINGUISTIC model, which incorporates linguistic constraints, outperformed the freer ALLPAIRS model. A large portion of this improvement came from English, a language for which the LINGUISTIC model greatly outperformed all other models evaluated. We found that the improved English analyses produced by the LINGUISTIC model were more consistent with this model's analyses of other languages. This consistency was not present for the English analyses produced by other models. We explore consistency in more detail in Section 5.

4.5 Comparison to Related Work

The likelihood models for both the strong monolingual baseline and the various multilingual mod-

els are the same, both expanding upon the standard DMV by adding coarse SHARED features. These coarse features, even in a monolingual setting, improved performance slightly over the weak baseline, perhaps by encouraging consistent treatment of the different finer-grained variants of parts-of-speech (Berg-Kirkpatrick et al., 2010).[2] The only difference between the multilingual systems and the strong baseline is whether or not cross-language influence is allowed through the prior.

While this progression of model structure is similar to that explored in Cohen and Smith (2009), Cohen and Smith saw their largest improvements from tying together parameters for the varieties of coarse parts-of-speech monolinugally, and then only moderate improvements from allowing cross-linguistic influence on top of monolingual sharing. When Cohen and Smith compared their best shared logistic-normal bilingual models to monolingual counter-parts for the languages they investigate (Chinese and English), they reported a relative error reduction of 5.3%. In comparison, with the LINGUISTIC model, we saw a much larger 16.9% relative error reduction over our strong baseline for these languages. Evaluating our LINGUISTIC model on the same test sets as (Cohen and Smith, 2009), sentences of length 10 or less in section 23 of PTB and sections 271-300 of CTB, we achieved an accuracy of 56.6 for Chinese and 60.3 for English. The best models of Cohen and Smith (2009) achieved accuracies of 52.0 and 62.0 respectively on these same test sets.

Our results indicate that the majority of our model's power beyond that of the standard DMV is derived from multilingual, and in particular, more-than-bilingual, interaction. These are, to the best of our knowledge, the first results of this kind for grammar induction without bitext.

5 Analysis

By examining the proposed parses we found that the LINGUISTIC and ALLPAIRS models produced analyses that were more consistent across languages than those of the other models. We also observed that the most common errors can be summarized succinctly by looking at attachment counts between coarse parts-of-speech. Figure 3 shows matrix representations of dependency counts. The area of a square is proportional to the number of order-collapsed dependencies where the column label is the head and the row label is the argument in the parses from each system. For ease of comprehension, we use the cross-lingual projections and only show counts for selected interesting classes.

Comparing Figure 3(c), which shows dependency counts proposed by the LINGUISTIC model, to Figure 3(a), which shows the same for the strong monolingual baseline, suggests that the analyses proposed by the LINGUISTIC model are more consistent across languages than are the analyses proposed by the monolingual model. For example, the monolingual learners are divided as to whether determiners or nouns head noun phrases. There is also confusion about which labels head whole sentences. Dutch has the problem that verbs modify pronouns more often than pronouns modify verbs, and pronouns are predicted to head sentences as often as verbs are. Spanish has some confusion about conjunctions, hypothesizing that verbs often attach to conjunctions, and conjunctions frequently head sentences. More subtly, the monolingual analyses are inconsistent in the way they head prepositional phrases. In the monolingual Portuguese hypotheses, prepositions modify nouns more often than nouns modify prepositions. In English, nouns modify prepositions, and prepositions modify verbs. Both the Dutch and Spanish models are ambivalent about the attachment of prepositions.

As has often been observed in other contexts (Liang et al., 2008), promoting agreement can improve accuracy in unsupervised learning. Not only are the analyses proposed by the LINGUISTIC model more consistent, they are also more in accordance with the gold analyses. Under the LINGUISTIC model, Dutch now attaches pronouns to verbs, and thus looks more like English, its sister in the phylogenetic tree. The LINGUISTIC model has also chosen consistent analyses for prepositional phrases and noun phrases, calling prepositions and nouns the heads of each, respectively. The problem of conjunctions heading Spanish sentences has also been corrected.

Figure 3(b) shows dependency counts for the GLOBAL multilingual model. Unsurprisingly, the analyses proposed under global constraint appear somewhat more consistent than those proposed under no multi-lingual constraint (now three lan-

[2] Coarse features that only tie nouns and verbs are explored in Berg-Kirkpatrick et al. (2010). We found that these were very effective for English and Chinese, but gave worse performance for other languages.

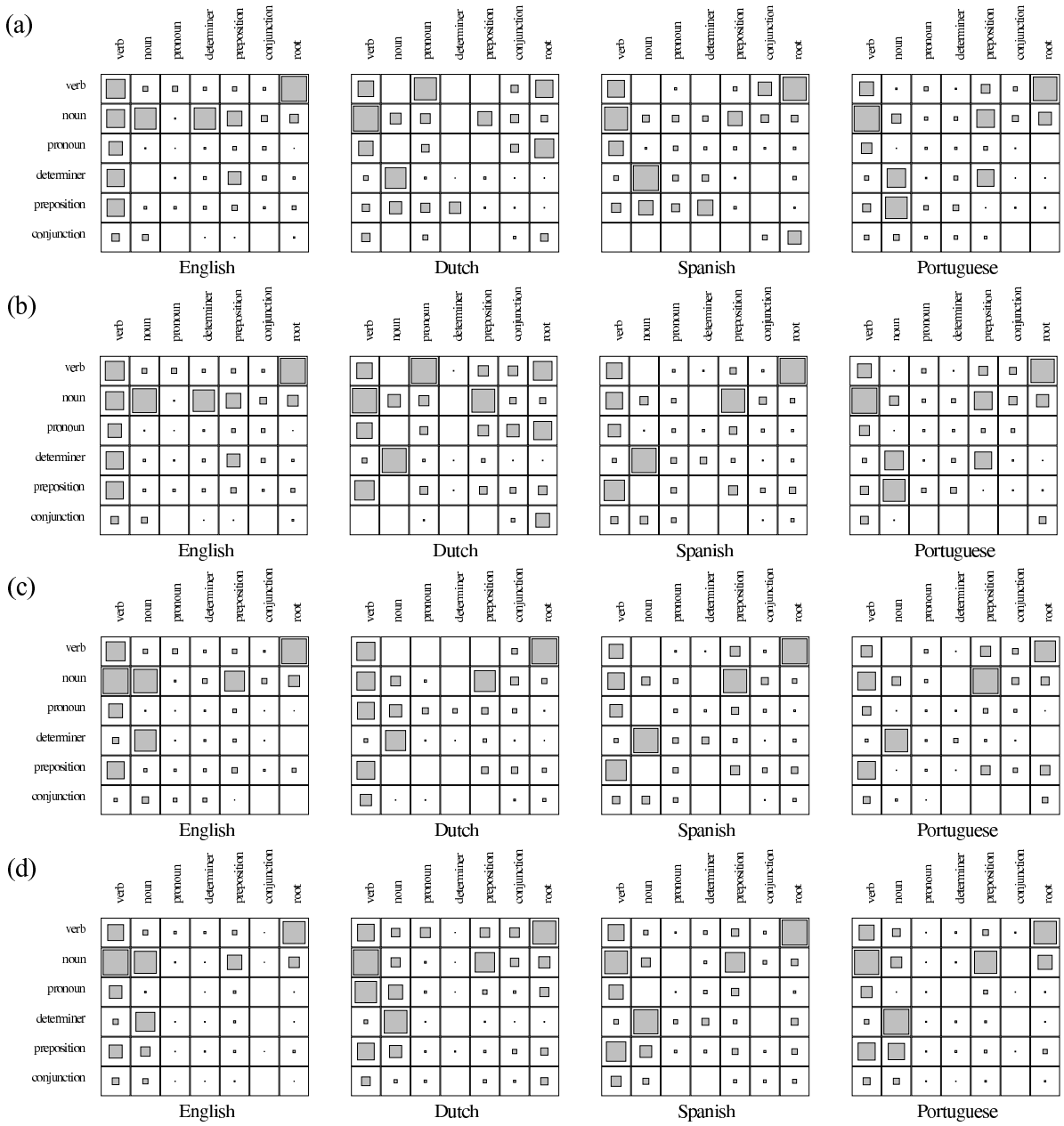


Figure 3: Dependency counts in proposed parses. Row label modifies column label. (a) Monolingual baseline with SHARED features. (b) GLOBAL model. (c) LINGUISTIC model. (d) Dependency counts in hand-labeled parses. Analyses proposed by monolingual baseline show significant inconsistencies across languages. Analyses proposed by LINGUISTIC model are more consistent across languages than those proposed by either the monolingual baseline or the GLOBAL model.

guages agree that prepositional phrases are headed by prepositions), but not as consistent as those proposed by the LINGUISTIC model.

Finally, Figure 3(d) shows dependency counts in the hand-labeled dependency parses. It appears that even the very consistent LINGUISTIC parses do not capture the non-determinism of prepositional phrase attachment to both nouns and verbs.

6 Conclusion

Even without translated texts, multilingual constraints expressed in the form of a phylogenetic prior on parameters can give substantial gains in grammar induction accuracy over treating languages in isolation. Additionally, articulated phylogenies that are sensitive to evolutionary structure can outperform not only limited flatter priors but also unconstrained all-pairs interactions.

7 Acknowledgements

This project is funded in part by the NSF under grant 0915265 and DARPA under grant N10AP20007.

References

T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *North American Chapter of the Association for Computational Linguistics*.

D. M. Bikel and D. Chiang. 2000. Two statistical parsing models applied to the Chinese treebank. In *Second Chinese Language Processing Workshop*.

A. Bouchard-Côté, P. Liang, D. Klein, and T. L. Griffiths. 2007. A probabilistic approach to diachronic phonology. In *Empirical Methods in Natural Language Processing*.

S. Buchholz and E. Marsi. 2006. Computational Natural Language Learning-X shared task on multilingual dependency parsing. In *Conference on Computational Natural Language Learning*.

D. Burkett and D. Klein. 2008. Two languages are better than one (for syntactic parsing). In *Empirical Methods in Natural Language Processing*.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *North American Chapter of the Association for Computational Linguistics*.

M. Collins. 1999. Head-driven statistical models for natural language parsing. In *Ph.D. thesis, University of Pennsylvania, Philadelphia*.

H. Daumé III. 2007. Frustratingly easy domain adaptation. In *Association for Computational Linguistics*.

J. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Association for Computational Linguistics*.

J. R. Finkel and C. D. Manning. 2009. Hierarchical bayesian domain adaptation. In *North American Chapter of the Association for Computational Linguistics*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Association for Computational Linguistics*.

J. Kuhn. 2004. Experiments in parallel-text based grammar induction. In *Association for Computational Linguistics*.

G. Kuzman, J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Association for Computational Linguistics/International Joint Conference on Natural Language Processing*.

P. Liang, D. Klein, and M. I. Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems*.

D. C. Liu, J. Nocedal, and C. Dong. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics*.

R. Salakhutdinov, S. Roweis, and Z. Ghahramani. 2003. Optimization with EM and expectation-conjugate-gradient. In *International Conference on Machine Learning*.

D. A. Smith and J. Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Empirical Methods in Natural Language Processing*.

B. Snyder, T. Naseem, and R. Barzilay. 2009a. Unsupervised multilingual grammar induction. In *Association for Computational Linguistics/International Joint Conference on Natural Language Processing*.

B. Snyder, T. Naseem, J. Eisenstein, and R. Barzilay. 2009b. Adding more languages improves unsupervised multilingual part-of-speech tagging: A Bayesian non-parametric approach. In *North American Chapter of the Association for Computational Linguistics*.

N. Xue, F-D Chiou, and M. Palmer. 2002. Building a large-scale annotated Chinese corpus. In *International Conference on Computational Linguistics*.

Improved Unsupervised POS Induction through Prototype Discovery

Omri Abend[1]* **Roi Reichart**[2] **Ari Rappoport**[1]

[1]Institute of Computer Science, [2]ICNC
Hebrew University of Jerusalem
{omria01|roiri|arir}@cs.huji.ac.il

Abstract

We present a novel fully unsupervised algorithm for POS induction from plain text, motivated by the cognitive notion of prototypes. The algorithm first identifies *landmark* clusters of words, serving as the cores of the induced POS categories. The rest of the words are subsequently mapped to these clusters. We utilize morphological and distributional representations computed in a fully unsupervised manner. We evaluate our algorithm on English and German, achieving the best reported results for this task.

1 Introduction

Part-of-speech (POS) tagging is a fundamental NLP task, used by a wide variety of applications. However, there is no single standard POS tagging scheme, even for English. Schemes vary significantly across corpora and even more so across languages, creating difficulties in using POS tags across domains and for multi-lingual systems (Jiang et al., 2009). Automatic induction of POS tags from plain text can greatly alleviate this problem, as well as eliminate the efforts incurred by manual annotations. It is also a problem of great theoretical interest. Consequently, POS induction is a vibrant research area (see Section 2).

In this paper we present an algorithm based on the theory of prototypes (Taylor, 2003), which posits that some members in cognitive categories are more central than others. These practically define the category, while the membership of other elements is based on their association with the central members. Our algorithm first clusters words based on a fine morphological representation. It then clusters the most frequent words, defining *landmark* clusters which constitute the cores of the categories. Finally, it maps the rest of the words to these categories. The last two stages utilize a distributional representation that has been shown to be effective for unsupervised parsing (Seginer, 2007).

We evaluated the algorithm in both English and German, using four different mapping-based and information theoretic clustering evaluation measures. The results obtained are generally better than all existing POS induction algorithms.

Section 2 reviews related work. Sections 3 and 4 detail the algorithm. Sections 5, 6 and 7 describe the evaluation, experimental setup and results.

2 Related Work

Unsupervised and semi-supervised POS tagging have been tackled using a variety of methods. Schütze (1995) applied latent semantic analysis. The best reported results (when taking into account all evaluation measures, see Section 5) are given by (Clark, 2003), which combines distributional and morphological information with the likelihood function of the Brown algorithm (Brown et al., 1992). Clark's tagger is very sensitive to its initialization. Reichart et al. (2010b) propose a method to identify the high quality runs of this algorithm. In this paper, we show that our algorithm outperforms not only Clark's mean performance, but often its best among 100 runs. Most research views the task as a sequential labeling problem, using HMMs (Merialdo, 1994; Banko and Moore, 2004; Wang and Schuurmans, 2005) and discriminative models (Smith and Eisner, 2005; Haghighi and Klein, 2006). Several

* Omri Abend is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1298–1307,
Uppsala, Sweden, 11-16 July 2010.

techniques were proposed to improve the HMM model. A Bayesian approach was employed by (Goldwater and Griffiths, 2007; Johnson, 2007; Gao and Johnson, 2008). Van Gael et al. (2009) used the infinite HMM with non-parametric priors. Graça et al. (2009) biased the model to induce a small number of possible tags for each word.

The idea of utilizing seeds and expanding them to less reliable data has been used in several papers. Haghighi and Klein (2006) use POS 'prototypes' that are manually provided and tailored to a particular POS tag set of a corpus. Freitag (2004) and Biemann (2006) induce an initial clustering and use it to train an HMM model. Dasgupta and Ng (2007) generate morphological clusters and use them to bootstrap a distributional model. Goldberg et al. (2008) use linguistic considerations for choosing a good starting point for the EM algorithm. Zhao and Marcus (2009) expand a partial dictionary and use it to learn disambiguation rules. Their evaluation is only at the type level and only for half of the words. Ravi and Knight (2009) use a dictionary and an MDL-inspired modification to the EM algorithm.

Many of these works use a dictionary providing allowable tags for each or some of the words. While this scenario might reduce human annotation efforts, it does not induce a tagging scheme but remains tied to an existing one. It is further criticized in (Goldwater and Griffiths, 2007).

Morphological representation. Many POS induction models utilize morphology to some extent. Some use simplistic representations of terminal letter sequences (e.g., (Smith and Eisner, 2005; Haghighi and Klein, 2006)). Clark (2003) models the entire letter sequence as an HMM and uses it to define a morphological prior. Dasgupta and Ng (2007) use the output of the *Morfessor* segmentation algorithm for their morphological representation. *Morfessor* (Creutz and Lagus, 2005), which we use here as well, is an unsupervised algorithm that segments words and classifies each segment as being a stem or an affix. It has been tested on several languages with strong results.

Our work has several unique aspects. First, our clustering method discovers prototypes in a fully unsupervised manner, mapping the rest of the words according to their association with the prototypes. Second, we use a distributional representation which has been shown to be effective for unsupervised parsing (Seginer, 2007). Third, we use a morphological representation based on signatures, which are sets of affixes that represent a family of words sharing an inflectional or derivational morphology (Goldsmith, 2001).

3 Distributional Algorithm

Our algorithm is given a plain text corpus and optionally a desired number of clusters k. Its output is a partitioning of words into clusters. The algorithm utilizes two representations, distributional and morphological. Although eventually the latter is used before the former, for clarity of presentation we begin by detailing the base distributional algorithm. In the next section we describe the morphological representation and its integration into the base algorithm.

Overview. The algorithm consists of two main stages: landmark clusters discovery, and word mapping. For the former, we first compute a distributional representation for each word. We then cluster the coordinates corresponding to high frequency words. Finally, we define *landmark clusters*. In the word mapping stage we map each word to the most similar landmark cluster.

The rationale behind using only the high frequency words in the first stage is twofold. First, prototypical members of a category are frequent (Taylor, 2003), and therefore we can expect the salient POS tags to be represented in this small subset. Second, higher frequency implies more reliable statistics. Since this stage determines the cores of all resulting clusters, it should be as accurate as possible.

Distributional representation. We use a simplified form of the elegant representation of lexical entries used by the Seginer unsupervised parser (Seginer, 2007). Since a POS tag reflects the grammatical role of the word and since this representation is effective to parsing, we were motivated to apply it to the present task.

Let W be the set of word types in the corpus. The right context entry of a word $x \in W$ is a pair of mappings $r\_int_x : W \rightarrow [0,1]$ and $r\_adj_x : W \rightarrow [0,1]$. For each $w \in W$, $r\_adj_x(w)$ is an adjacency score of w to x, reflecting w's tendency to appear on the right hand side of x.

For each $w \in W$, $r\_int_x(w)$ is an interchangeability score of x with w, reflecting the tendency of w to appear to the left of words that tend to appear to the right of x. This can be viewed as a

similarity measure between words with respect to their right context. The higher the scores the more the words tend to be adjacent/interchangeable.

Left context parameters l_int_x and l_adj_x are defined analogously.

There are important subtleties in these definitions. First, for two words $x, w \in W$, $r\_adj_x(w)$ is generally different from $l\_adj_w(x)$. For example, if w is a high frequency word and x is a low frequency word, it is likely that w appears many times to the right of x, yielding a high $r\_adj_x(w)$, but that x appears only a few times to the left of w yielding a low $l\_adj_w(x)$. Second, from the definition of $r\_int_x(w)$ and $r\_int_w(x)$, it is clear that they need not be equal.

These functions are computed incrementally by a bootstrapping process. We initialize all mappings to be identically 0. We iterate over the words in the training corpus. For every word instance x, we take the word immediately to its right y and update x's right context using y's left context:

$$\forall w \in W: \quad r\_int_x(w) \mathrel{+}= \frac{l\_adj_y(w)}{N(y)}$$

$$\forall w \in W: \quad r\_adj_x(w) \mathrel{+}= \begin{cases} 1 & w = y \\ \frac{l\_int_y(w)}{N(y)} & w \neq y \end{cases}$$

The division by $N(y)$ (the number of times y appears in the corpus before the update) is done in order not to give a disproportional weight to high frequency words. Also, $r\_int_x(w)$ and $r\_adj_x(w)$ might become larger than 1. We therefore normalize them after all updates are performed by the number of occurrences of x in the corpus.

We update l_int_x and l_adj_x analogously using the word z immediately to the left of x. The updates of the left and right functions are done in parallel.

We define the distributional representation of a word type x to be a $4|W| + 2$ dimensional vector v_x. Each word w yields four coordinates, one for each direction (left/right) and one for each mapping type (int/adj). Two additional coordinates represent the frequency in which the word appears to the left and to the right of a stopping punctuation. Of the $4|W|$ coordinates corresponding to words, we allow only $2n$ to be non-zero: the n top scoring among the right side coordinates (those of r_int_x and r_adj_x), and the n top scoring among the left side coordinates (those of l_int_x and l_adj_x). We used $n = 50$.

The distance between two words is defined to be one minus the cosine of the angle between their representation vectors.

Coordinate clustering. Each of our landmark clusters will correspond to a set of high frequency words (HFWs). The number of HFWs is much larger than the number of expected POS tags. Hence we should cluster HFWs. Our algorithm does that by unifying some of the non-zero coordinates corresponding to HFWs in the distributional representation defined above.

We extract the words that appear more than N times per million[1] and apply the following procedure I times (5 in our experiments).

We run average link clustering with a threshold α (AVGLINK$_\alpha$, (Jain et al., 1999)) on these words, in each iteration initializing every HFW to have its own cluster. AVGLINK$_\alpha$ means running the average link algorithm until the two closest clusters have a distance larger than α. We then use the induced clustering to update the distributional representation, by collapsing all coordinates corresponding to words appearing in the same cluster into a single coordinate whose value is the sum of the collapsed coordinates' values. In order to produce a conservative (fine) clustering, we used a relatively low α value of 0.25.

Note that the AVGLINK$_\alpha$ initialization in each of the I iterations assigns each HFW to a separate cluster. The iterations differ in the distributional representation of the HFWs, resulting from the previous iterations.

In our English experiments, this process reduced the dimension of the HFWs set (the number of coordinates that are non-zero in at least one of the HFWs) from 14365 to 10722. The average number of non-zero coordinates per word decreased from 102 to 55.

Since all eventual POS categories correspond to clusters produced at this stage, to reduce noise we delete clusters of less than five elements.

Landmark detection. We define landmark clusters using the clustering obtained in the final iteration of the coordinate clustering stage. However, the number of clusters might be greater than the desired number k, which is an optional parameter of the algorithm. In this case we select a subset of k clusters that best covers the HFW space. We use the following heuristic. We start from the most frequent cluster, and greedily select the clus-

[1] We used $N = 100$, yielding 1242 words for English and 613 words for German.

ter farthest from the clusters already selected. The distance between two clusters is defined to be the average distance between their members. A cluster's distance from a set of clusters is defined to be its minimal distance from the clusters in the set. The final set of clusters $\{L_1, ..., L_k\}$ and their members are referred to as *landmark clusters* and *prototypes*, respectively.

Mapping all words. Each word $w \in W$ is assigned the cluster L_i that contains its nearest prototype:

$$d(w, L_i) = min_{x \in L_i}\{1 - cos(v_w, v_x)\}$$
$$Map(w) = argmin_{L_i}\{d(w, L_i)\}$$

Words that appear less than 5 times are considered as *unknown words*. We consider two schemes for handling unknown words. One randomly maps each such word to a cluster, using a probability proportional to the number of unique known words already assigned to that cluster. However, when the number k of landmark clusters is relatively large, it is beneficial to assign all unknown words to a separate new cluster (after running the algorithm with $k - 1$). In our experiments, we use the first option when k is below some threshold (we used 15), otherwise we use the second.

4 Morphological Model

The morphological model generates another word clustering, based on the notion of a signature. This clustering is integrated with the distributional model as described below.

4.1 Morphological Representation

We use the *Morfessor* (Creutz and Lagus, 2005) word segmentation algorithm. First, all words in the corpus are segmented. Then, for each stem, the set of all affixes with which it appears (its *signature*, (Goldsmith, 2001)) is collected. The morphological representation of a word type is then defined to be its stem's signature in conjunction with its specific affixes[2] (See Figure 1).

We now collect all words having the same representation. For instance, if the words *joined* and *painted* are found to have the same signature, they would share the same cluster since both have the affix '_ed'. The word *joins* does not share the same cluster with them since it has a different affix, '_s'. This results in coarse-grained clusters exclusively defined according to morphology.

[2] A word may contain more than a single affix.

Types	join	joins	joined	joining
Stem	join	join	join	join
Affixes	ϕ	_s	_ed	_ing
Signature	{ϕ,_ed,_s,_ing}			

Figure 1: An example for a morphological representation, defined to be the conjunction of its affix(es) with the stem's signature.

In addition, we incorporate capitalization information into the model, by constraining all words that appear capitalized in more than half of their instances to belong to a separate cluster, regardless of their morphological representation. The motivation for doing so is practical: capitalization is used in many languages to mark grammatical categories. For instance, in English capitalization marks the category of proper names and in German it marks the noun category . We report English results both with and without this modification.

Words that contain non-alphanumeric characters are represented as the sequence of the non-alphanumeric characters they include, e.g., 'vis-à-vis' is represented as *("-", "-")*. We do not assign a morphological representation to words including more than one stem (like *weatherman*), to words that have a null affix (i.e., where the word is identical to its stem) and to words whose stem is not shared by any other word (signature of size 1). Words that were not assigned a morphological representation are included as singletons in the morphological clustering.

4.2 Distributional-Morphological Algorithm

We detail the modifications made to our base distributional algorithm given the morphological clustering defined above.

Coordinate clustering and landmarks. We constrain AVGLINK$_\alpha$ to begin by forming links between words appearing in the same morphological cluster. Only when the distance between the two closest clusters gets above α we remove this constraint and proceed as before. This is equivalent to performing AVGLINK$_\alpha$ separately within each morphological cluster and then using the result as an initial condition for an AVGLINK$_\alpha$ coordinate clustering. The modified algorithm in this stage is otherwise identical to the distributional algorithm.

Word mapping. In this stage words that are not prototypes are mapped to one of the landmark

clusters. A reasonable strategy would be to map all words sharing a morphological cluster as a single unit. However, these clusters are too coarse-grained. We therefore begin by partitioning the morphological clusters into sub-clusters according to their distributional behavior. We do so by applying AVGLINK$_\beta$ (the same as AVGLINK$_\alpha$ but with a different parameter) to each morphological cluster. Since our goal is cluster *refinement*, we use a β that is considerably higher than α (0.9).

We then find the closest prototype to each such sub-cluster (averaging the distance across all of the latter's members) and map it as a single unit to the cluster containing that prototype.

5 Clustering Evaluation

We evaluate the clustering produced by our algorithm using an external quality measure: we take a corpus tagged by gold standard tags, tag it using the induced tags, and compare the two taggings. There is no single accepted measure quantifying the similarity between two taggings. In order to be as thorough as possible, we report results using four known measures, two mapping-based measures and two information theoretic ones.

Mapping-based measures. The induced clusters have arbitrary names. We define two mapping schemes between them and the gold clusters. After the induced clusters are mapped, we can compute a derived accuracy. The **Many-to-1** measure finds the mapping between the gold standard clusters and the induced clusters which maximizes accuracy, allowing several induced clusters to be mapped to the same gold standard cluster. The **1-to-1** measure finds the mapping between the induced and gold standard clusters which maximizes accuracy such that no two induced clusters are mapped to the same gold cluster. Computing this mapping is equivalent to finding the maximal weighted matching in a bipartite graph, whose weights are given by the intersection sizes between matched classes/clusters. As in (Reichart and Rappoport, 2008), we use the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957) to solve this problem.

Information theoretic measures. These are based on the observation that a good clustering reduces the uncertainty of the gold tag given the induced cluster, and vice-versa. Several such measures exist; we use **V** (Rosenberg and Hirschberg, 2007) and **NVI** (Reichart and Rappoport, 2009), **VI**'s (Meila, 2007) normalized version.

6 Experimental Setup

Since a goal of unsupervised POS tagging is inducing an annotation scheme, comparison to an existing scheme is problematic. To address this problem we compare to three different schemes in two languages. In addition, the two English schemes we compare with were designed to tag corpora contained in our training set, and have been widely and successfully used with these corpora by a large number of applications.

Our algorithm was run with the exact same parameters on both languages: $N = 100$ (high frequency threshold), $n = 50$ (the parameter that determines the effective number of coordinates), $\alpha = 0.25$ (cluster separation during landmark cluster generation), $\beta = 0.9$ (cluster separation during refinement of morphological clusters).

The algorithm we compare with in most detail is (Clark, 2003), which reports the best current results for this problem (see Section 7). Since Clark's algorithm is sensitive to its initialization, we ran it a 100 times and report its average and standard deviation in each of the four measures. In addition, we report the percentile in which our result falls with respect to these 100 runs.

Punctuation marks are very frequent in corpora and are easy to cluster. As a result, including them in the evaluation greatly inflates the scores. For this reason we do not assign a cluster to punctuation marks and we report results using this policy, which we recommend for future work. However, to be able to directly compare with previous work, we also report results for the full POS tag set. We do so by assigning a singleton cluster to each punctuation mark (in addition to the k required clusters). This simple heuristic yields very high performance on punctuation, scoring (when all other words are assumed perfect tagging) 99.6% (99.1%) 1-to-1 accuracy when evaluated against the English fine (coarse) POS tag sets, and 97.2% when evaluated against the German POS tag set.

For English, we trained our model on the 39832 sentences which constitute sections 2-21 of the PTB-WSJ and on the 500K sentences from the NYT section of the NANC newswire corpus (Graff, 1995). We report results on the WSJ part of our data, which includes 950028 words tokens in 44389 types. Of the tokens, 832629 (87.6%)

English	Fine k=13				Coarse k=13				Fine k=34			
	Prototype Tagger	Clark μ	Clark σ	%	Prototype Tagger	Clark μ	Clark σ	%	Prototype Tagger	Clark μ	Clark σ	%
Many–to–1	**61.0**	55.1	1.6	100	**70.0**	66.9	2.1	94	**71.6**	69.8	1.5	90
	55.5	48.8	1.8	100	**66.1**	62.6	2.3	94	**67.5**	65.5	1.7	90
1–to–1	**60.0**	52.2	1.9	100	**58.1**	49.4	2.9	100	**63.5**	54.5	1.6	100
	54.9	46.0	2.2	100	**53.7**	43.8	3.3	100	**58.8**	48.5	1.8	100
NVI	**0.652**	0.773	0.027	100	**0.841**	0.972	0.036	100	**0.663**	0.725	0.018	100
	0.795	0.943	0.033	100	**1.052**	1.221	0.046	100	**0.809**	0.885	0.022	100
V	**0.636**	0.581	0.015	100	**0.590**	0.543	0.018	100	**0.677**	0.659	0.008	100
	0.542	0.478	0.019	100	**0.484**	0.429	0.023	100	**0.608**	0.588	0.010	98

German	k=17				k=26			
	Prototype Tagger	Clark μ	Clark σ	%	Prototype Tagger	Clark μ	Clark σ	%
Many–to-1	64.6	**64.7**	1.2	41	**68.2**	67.8	1.0	60
	58.9	**59.1**	1.4	40	**63.2**	62.8	1.2	60
1–to–1	**53.7**	52.0	1.8	77	**56.0**	52.0	2.1	99
	48.0	46.0	2.3	78	**50.7**	45.9	2.6	99
NVI	**0.667**	0.675	0.019	66	**0.640**	0.682	0.019	100
	0.819	0.829	0.025	66	**0.785**	0.839	0.025	100
V	**0.646**	0.645	0.010	50	**0.675**	0.657	0.008	100
	0.552	**0.553**	0.013	48	**0.596**	0.574	0.010	100

Table 1: Top: English. Bottom: German. Results are reported for our model (Prototype Tagger), Clark's average score (μ), Clark's standard deviation (σ) and the fraction of Clark's results that scored worse than our model (%). For the mapping based measures, results are accuracy percentage. For $V \in [0,1]$, higher is better. For high quality output, $NVI \in [0,1]$ as well, and lower is better. In each entry, the top number indicates the score when including punctuation and the bottom number the score when excluding it. In English, our results are always better than Clark's. In German, they are almost always better.

are not punctuation. The percentage of unknown words (those appearing less than five times) is 1.6%. There are 45 clusters in this annotation scheme, 34 of which are not punctuation.

We ran each algorithm both with k=13 and k=34 (the number of desired clusters). We compare the output to two annotation schemes: the fine grained PTB WSJ scheme, and the coarse grained tags defined in (Smith and Eisner, 2005). The output of the k=13 run is evaluated both against the coarse POS tag annotation (the *'Coarse k=13'* scenario) and against the full PTB-WSJ annotation scheme (the *'Fine k=13'* scenario). The k=34 run is evaluated against the full PTB-WSJ annotation scheme (the *'Fine k=34'* scenario).

The POS cluster frequency distribution tends to be skewed: each of the 13 most frequent clusters in the PTB-WSJ cover more than 2.5% of the tokens (excluding punctuation) and together 86.3% of them. We therefore chose k=13, since it is both the number of coarse POS tags (excluding punctuation) as well as the number of frequent POS tags in the PTB-WSJ annotation scheme. We chose k=34 in order to evaluate against the full 34 tags PTB-WSJ annotation scheme (excluding punctuation) using the same number of clusters.

For German, we trained our model on the 20296 sentences of the NEGRA corpus (Brants, 1997) and on the first 450K sentences of the DeWAC corpus (Baroni et al., 2009). DeWAC is a corpus extracted by web crawling and is therefore out of domain. We report results on the NEGRA part, which includes 346320 word tokens of 49402 types. Of the tokens, 289268 (83.5%) are not punctuation. The percentage of unknown words (those appearing less than five times) is 8.1%. There are 62 clusters in this annotation scheme, 51 of which are not punctuation.

We ran the algorithms with k=17 and k=26. k=26 was chosen since it is the number of clusters that cover each more than 0.5% of the NEGRA tokens, and in total cover 96% of the (non-punctuation) tokens. In order to test our algorithm in another scenario, we conducted experiments with k=17 as well, which covers 89.9% of the tokens. All outputs are compared against NEGRA's gold standard scheme.

We do not report results for k=51 (where the number of gold clusters is the same as the number of induced clusters), since our algorithm produced only 42 clusters in the landmark detection stage. We could of course have modified the parameters to allow our algorithm to produce 51 clusters. However, we wanted to use the exact same parameters as those used for the English experiments to minimize the issue of parameter tuning.

In addition to the comparisons described above, we present results of experiments (in the 'Fine

	B	B+M	B+C	F(I=1)	F
M-to-1	53.3	54.8	58.2	57.3	**61.0**
1-to-1	50.2	51.7	55.1	54.8	**60.0**
NVI	0.782	0.720	0.710	0.742	**0.652**
V	0.569	0.598	0.615	0.597	**0.636**

Table 2: A comparison of partial versions of the model in the 'Fine k=13' WSJ scenario. M-to-1 and 1-to-1 results are reported in accuracy percentage. Lower NVI is better. *B* is the strictly distributional algorithm, *B+M* adds the morphological model, *B+C* adds capitalization to *B*, *F(I=1)* consists of all components, where only one iteration of coordinate clustering is performed, and *F* is the full model.

	M-to-1	1-to-1	V	VI
Prototype	**71.6**	**63.5**	**0.677**	**2.00**
Clark	69.8	54.5	0.659	2.18
HK	–	41.3	–	–
J	43–62	37–47	–	4.23–5.74
GG	–	–	–	2.8
GJ	–	40–49.9	–	4.03–4.47
VG	–	–	0.54-0.59	2.5–2.9
GGTP-45	65.4	44.5	–	–
GGTP-17	70.2	49.5	–	–

Table 4: Comparison of our algorithms with the recent fully unsupervised POS taggers for which results are reported. The models differ in the annotation scheme, the corpus size and the number of induced clusters (k) that they used. HK: (Haghighi and Klein, 2006), 193K tokens, fine tags, k=45. GG: (Goldwater and Griffiths, 2007), 24K tokens, coarse tags, k=17. J : (Johnson, 2007), 1.17M tokens, fine tags, k=25–50. GJ: (Gao and Johnson, 2008), 1.17M tokens, fine tags, k=50. VG: (Van Gael et al., 2009), 1.17M tokens, fine tags, k=47–192. GGTP-45: (Graça et al., 2009), 1.17M tokens, fine tags, k=45. GGTP-17: (Graça et al., 2009), 1.17M tokens, coarse tags, k=17. Lower VI values indicate better clustering. VI is computed using e as the base of the logarithm. Our algorithm gives the best results.

k=13' scenario) that quantify the contribution of each component of the algorithm. We ran the base distributional algorithm, a variant which uses only capitalization information (i.e., has only one non-singleton morphological class, that of words appearing capitalized in most of their instances) and a variant which uses no capitalization information, defining the morphological clusters according to the morphological representation alone.

7 Results

Table 1 presents results for the English and German experiments. For English, our algorithm obtains better results than Clark's in all measures and scenarios. It is without exception better than the average score of Clark's and in most cases better than the maximal Clark score obtained in 100 runs.

A significant difference between our algorithm and Clark's is that the latter, like most algorithms which addressed the task, induces the clustering

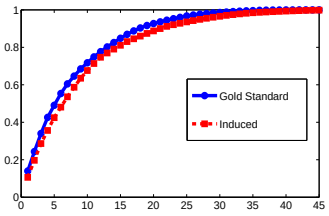


Figure 2: POS class frequency distribution for our model and the gold standard, in the 'Fine k=34' scenario. The distributions are similar.

by maximizing a non-convex function. These functions have many local maxima and the specific solution to which algorithms that maximize them converge strongly depends on their (random) initialization. Therefore, their output's quality often significantly diverges from the average. This issue is discussed in depth in (Reichart et al., 2010b). Our algorithm is deterministic[3].

For German, in the k=26 scenario our algorithm outperforms Clark's, often outperforming even its maximum in 100 runs. In the k=17 scenario, our algorithm obtains a higher score than Clark with probability 0.4 to 0.78, depending on the measure and scenario. Clark's average score is slightly better in the Many-to-1 measure, while our algorithm performs somewhat better than Clark's average in the 1-to-1 and NVI measures.

The DeWAC corpus from which we extracted statistics for the German experiments is out of domain with respect to NEGRA. The corresponding corpus in English, NANC, is a newswire corpus and therefore clearly in-domain with respect to WSJ. This is reflected by the percentage of unknown words, which was much higher in German than in English (8.1% and 1.6%), lowering results.

Table 2 shows the effect of each of our algorithm's components. Each component provides an improvement over the base distributional algorithm. The full coordinate clustering stage (several iterations, F) considerably improves the score over a single iteration (F(I=1)). Capitalization information increases the score more than the morphological information, which might stem from the granularity of the POS tag set with respect to names. This analysis is supported by similar experiments we made in the 'Coarse k=13' scenario (not shown in tables here). There, the decrease in performance was only of 1%–2% in the mapping

[3]The fluctuations inflicted on our algorithm by the random mapping of unknown words are of less than 0.1% .

	Excluding Punctuation				Including Punctuation				Perfect Punctuation			
	M-to-1	1-to-1	NVI	V	M-to-1	1-to-1	NVI	V	M-to-1	1-to-1	NVI	V
Van Gael	59.1	48.4	0.999	0.530	62.3	51.3	0.861	0.591	64.0	54.6	0.820	0.610
Prototype	**67.5**	**58.8**	**0.809**	**0.608**	**71.6**	**63.5**	**0.663**	**0.677**	**71.6**	**63.9**	**0.659**	**0.679**

Table 3: Comparison between the *iHMM: PY-fixed* model (Van Gael et al., 2009) and ours with various punctuation assignment schemes. Left section: punctuation tokens are excluded. Middle section: punctuation tokens are included. Right section: perfect assignment of punctuation is assumed.

based measures and 3.5% in the V measure.

Finally, Table 4 presents reported results for all recent algorithms we are aware of that tackled the task of unsupervised POS induction from plain text. Results for our algorithm's and Clark's are reported for the 'Fine, k=34' scenario. The settings of the various experiments vary in terms of the exact annotation scheme used (coarse or fine grained) and the size of the test set. However, the score differences are sufficiently large to justify the claim that our algorithm is currently the best performing algorithm on the PTB-WSJ corpus for POS induction from plain text[4].

Since previous works provided results only for the scenario in which punctuation is included, the reported results are not directly comparable. In order to quantify the effect various punctuation schemes have on the results, we evaluated the *'iHMM: PY-fixed'* model (Van Gael et al., 2009) and ours when punctuation is excluded, included or perfectly tagged[5]. The results (Table 3) indicate that most probably even after an appropriate correction for punctuation, our model remains the best performing one.

8 Discussion

In this work we presented a novel unsupervised algorithm for POS induction from plain text. The algorithm first generates relatively accurate clusters of high frequency words, which are subsequently used to bootstrap the entire clustering. The distributional and morphological representations that we use are novel for this task.

We experimented on two languages with mapping and information theoretic clustering evaluation measures. Our algorithm obtains the best reported results on the English PTB-WSJ corpus. In addition, our results are almost always better than Clark's on the German NEGRA corpus.

We have also performed a manual error analysis, which showed that our algorithm performs much better on closed classes than on open classes. In order to asses this quantitatively, let us define a random variable for each of the gold clusters, which receives a value corresponding to each induced cluster with probability proportional to their intersection size. For each gold cluster, we compute the entropy of this variable. In addition, we greedily map each induced cluster to a gold cluster and compute the ratio between their intersection size and the size of the gold cluster (mapping accuracy).

We experimented in the 'Fine k=34' scenario. The clusters that obtained the best scores were (brackets indicate mapping accuracy and entropy for each of these clusters) coordinating conjunctions (95%, 0.32), prepositions (94%, 0.32), determiners (94%, 0.44) and modals (93%, 0.45). These are all closed classes.

The classes on which our algorithm performed worst consist of open classes, mostly verb types: past tense verbs (47%, 2.2), past participle verbs (44%, 2.32) and the morphologically unmarked non-3rd person singular present verbs (32%, 2.86). Another class with low performance is the proper nouns (37%, 2.9). The errors there are mostly of three types: confusions between common and proper nouns (sometimes due to ambiguity), unknown words which were put in the unknown words cluster, and abbreviations which were given a separate class by our algorithm. Finally, the algorithm's performance on the heterogeneous adverbs class (19%, 3.73) is the lowest.

Clark's algorithm exhibits[6] a similar pattern with respect to open and closed classes. While his algorithm performs considerably better on adverbs (15% mapping accuracy difference and 0.71 entropy difference), our algorithm scores considerably better on prepositions (17%, 0.77), superlative adjectives (38%, 1.37) and plural proper names (45%, 1.26).

[4] Graça et al. (2009) report very good results for 17 tags in the M-1 measure. However, their 1-1 results are quite poor, and results for the common IT measures were not reported. Their results for 45 tags are considerably lower.

[5] We thank the authors for sending us their data.

[6] Using average mapping accuracy and entropy over the 100 runs.

Naturally, this analysis might reflect the arbitrary nature of a manually design POS tag set rather than deficiencies in automatic POS induction algorithms. In future work we intend to analyze the output of such algorithms in order to improve POS tag sets.

Our algorithm and Clark's are monosemous (i.e., they assign each word exactly one tag), while most other algorithms are polysemous. In order to assess the performance loss caused by the monosemous nature of our algorithm, we took the M-1 greedy mapping computed for the entire dataset and used it to compute accuracy over the monosemous and polysemous words separately. Results are reported for the English 'Fine k=34' scenario (without punctuation). We define a word to be monosemous if more than 95% of its tokens are assigned the same gold standard tag. For English, there are approximately 255K polysemous tokens and 578K monosemous ones. As expected, our algorithm is much more accurate on the monosemous tokens, achieving 76.6% accuracy, compared to 47.1% on the polysemous tokens.

The evaluation in this paper is done at the token level. Type level evaluation, reflecting the algorithm's ability to detect the set of possible POS tags for each word type, is important as well. It could be expected that a monosemous algorithm such as ours would perform poorly in a type level evaluation. In (Reichart et al., 2010a) we discuss type level evaluation at depth and propose type level evaluation measures applicable to the POS induction problem. In that paper we compare the performance of our Prototype Tagger with leading unsupervised POS tagging algorithms (Clark, 2003; Goldwater and Griffiths, 2007; Gao and Johnson, 2008; Van Gael et al., 2009). Our algorithm obtained the best results in 4 of the 6 measures in a margin of 4–6%, and was second best in the other two measures. Our results were better than Clark's (the only other monosemous algorithm evaluated there) on all measures in a margin of 5–21%. The fact that our monosemous algorithm was better than good polysemous algorithms in a type level evaluation can be explained by the prototypical nature of the POS phenomenon (a longer discussion is given in (Reichart et al., 2010a)). However, the quality upper bound for monosemous algorithms is obviously much lower than that for polysemous algorithms, and we expect polysemous algorithms to outperform monosemous algorithms in the future in both type level and token level evaluations.

The skewed (Zipfian) distribution of POS class frequencies in corpora is a problem for many POS induction algorithms, which by default tend to induce a clustering having a balanced distribution. Explicit modifications to these algorithms were introduced in order to bias their model to produce such a distribution (see (Clark, 2003; Johnson, 2007; Reichart et al., 2010b)). An appealing property of our model is its ability to induce a skewed distribution without being explicitly tuned to do so, as seen in Figure 2.

Acknowledgements. We would like to thank Yoav Seginer for his help with his parser.

References

Michele Banko and Robert C. Moore, 2004. *Part of Speech Tagging in Context.* COLING '04.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi and Eros Zanchetta, 2009. *The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora.* Language Resources and Evaluation.

Chris Biemann, 2006. *Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering.* COLING-ACL '06 Student Research Workshop.

Thorsten Brants, 1997. *The NEGRA Export Format.* CLAUS Report, Saarland University.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souze, Jenifer C. Lai and Robert Mercer, 1992. *Class-Based N-Gram Models of Natural Language.* Computational Linguistics, 18(4):467–479.

Alexander Clark, 2003. *Combining Distributional and Morphological Information for Part of Speech Induction.* EACL '03.

Mathias Creutz and Krista Lagus, 2005. *Inducing the Morphological Lexicon of a Natural Language from Unannotated Text.* AKRR '05.

Sajib Dasgupta and Vincent Ng, 2007. *Unsupervised Part-of-Speech Acquisition for Resource-Scarce Languages.* EMNLP-CoNLL '07.

Dayne Freitag, 2004. *Toward Unsupervised Whole-Corpus Tagging.* COLING '04.

Jianfeng Gao and Mark Johnson, 2008. *A Comparison of Bayesian Estimators for Unsupervised Hidden Markov Model POS Taggers.* EMNLP '08.

Yoav Goldberg, Meni Adler and Michael Elhadad, 2008. *EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start).* ACL '08.

John Goldsmith, 2001. *Unsupervised Learning of the Morphology of a Natural Language.* Computational Linguistics, 27(2):153–198.

Sharon Goldwater and Tom Griffiths, 2007. *Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging.* ACL '07.

João Graça, Kuzman Ganchev, Ben Taskar and Fernando Pereira, 2009. Posterior vs. Parameter Sparsity in Latent Variable Models. *NIPS '09.*

David Graff, 1995. *North American News Text Corpus.* Linguistic Data Consortium. LDC95T21.

Aria Haghighi and Dan Klein, 2006. *Prototype-driven Learning for Sequence Labeling.* HLT–NAACL '06.

Anil K. Jain, Narasimha M. Murty and Patrick J. Flynn, 1999. *Data Clustering: A Review.* ACM Computing Surveys 31(3):264–323.

Wenbin Jiang, Liang Huang and Qun Liu, 2009. *Automatic Adaptation of Annotation Standards: Chinese Word Segmentation and POS Tagging – A Case Study.* ACL '09.

Mark Johnson, 2007. *Why Doesnt EM Find Good HMM POS-Taggers?* EMNLP-CoNLL '07.

Harold W. Kuhn, 1955. *The Hungarian method for the Assignment Problem.* Naval Research Logistics Quarterly, 2:83-97.

Marina Meila, 2007. *Comparing Clustering – an Information Based Distance.* Journal of Multivariate Analysis, 98:873–895.

Bernard Merialdo, 1994. *Tagging English Text with a Probabilistic Model.* Computational Linguistics, 20(2):155–172.

James Munkres, 1957. *Algorithms for the Assignment and Transportation Problems.* Journal of the SIAM, 5(1):32–38.

Sujith Ravi and Kevin Knight, 2009. *Minimized Models for Unsupervised Part-of-Speech Tagging.* ACL '09.

Roi Reichart and Ari Rappoport, 2008. *Unsupervised Induction of Labeled Parse Trees by Clustering with Syntactic Features.* COLING '08.

Roi Reichart and Ari Rappoport, 2009. *The NVI Clustering Evaluation Measure.* CoNLL '09.

Roi Reichart, Omri Abend and Ari Rappoport, 2010a. *Type Level Clustering Evaluation: New Measures and a POS Induction Case Study.* CoNLL '10.

Roi Reichart, Raanan Fattal and Ari Rappoport, 2010b. *Improved Unsupervised POS Induction Using Intrinsic Clustering Quality and a Zipfian Constraint.* CoNLL '10.

Andrew Rosenberg and Julia Hirschberg, 2007. *V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.* EMNLP '07.

Hinrich Schütze, 1995. *Distributional part-of-speech tagging.* EACL '95.

Yoav Seginer, 2007. *Fast Unsupervised Incremental Parsing.* ACL '07.

Noah A. Smith and Jason Eisner, 2005. *Contrastive Estimation: Training Log-Linear Models on Unlabeled Data.* ACL '05.

John R. Taylor, 2003. *Linguistic Categorization: Prototypes in Linguistic Theory, Third Edition.* Oxford University Press.

Jurgen Van Gael, Andreas Vlachos and Zoubin Ghahramani, 2009. *The Infinite HMM for Unsupervised POS Tagging.* EMNLP '09.

Qin Iris Wang and Dale Schuurmans, 2005. *Improved Estimation for Unsupervised Part-of-Speech Tagging.* IEEE NLP–KE '05.

Qiuye Zhao and Mitch Marcus, 2009. *A Simple Unsupervised Learner for POS Disambiguation Rules Given Only a Minimal Lexicon.* EMNLP '09.

Extraction and Approximation of Numerical Attributes from the Web

Dmitry Davidov
ICNC
The Hebrew University
Jerusalem, Israel
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University
Jerusalem, Israel
arir@cs.huji.ac.il

Abstract

We present a novel framework for automated extraction and approximation of numerical object attributes such as height and weight from the Web. Given an object-attribute pair, we discover and analyze attribute information for a set of comparable objects in order to infer the desired value. This allows us to approximate the desired numerical values even when no exact values can be found in the text.

Our framework makes use of relation defining patterns and WordNet similarity information. First, we obtain from the Web and WordNet a list of terms similar to the given object. Then we retrieve attribute values for each term in this list, and information that allows us to compare different objects in the list and to infer the attribute value range. Finally, we combine the retrieved data for all terms from the list to select or approximate the requested value.

We evaluate our method using automated question answering, WordNet enrichment, and comparison with answers given in Wikipedia and by leading search engines. In all of these, our framework provides a significant improvement.

1 Introduction

Information on various numerical properties of physical objects, such as length, width and weight is fundamental in question answering frameworks and for answering search engine queries. While in some cases manual annotation of objects with numerical properties is possible, it is a hard and labor intensive task, and is impractical for dealing with the vast amount of objects of interest. Hence, there is a need for automated semantic acquisition algorithms targeting such properties.

In addition to answering direct questions, the ability to make a crude comparison or estimation of object attributes is important as well. For example, it allows to disambiguate relationships between objects such as *X part-of Y* or *X inside Y*. Thus, a coarse approximation of the height of a house and a window is sufficient to decide that in the 'house window' nominal compound, 'window' is very likely to be a part of house and not vice versa. Such relationship information can, in turn, help summarization, machine translation or textual entailment tasks.

Due to the importance of relationship and attribute acquisition in NLP, numerous methods were proposed for extraction of various lexical relationships and attributes from text. Some of these methods can be successfully used for extracting numerical attributes. However, numerical attribute extraction is substantially different in two aspects, verification and approximation.

First, unlike most general lexical attributes, numerical attribute values are comparable. It usually makes no sense to compare the names of two actors, but it is meaningful to compare their ages. The ability to compare values of different objects allows to improve attribute extraction precision by verifying consistency with attributes of other similar objects. For example, suppose that for Toyota Corolla width we found two different values, 1.695m and 27cm. The second value can be either an extraction error or a length of a toy car. Extracting and looking at width values for different car brands and for 'cars' in general we find:

- Boundaries: Maximal car width is 2.195m, minimal is 88cm.
- Average: Estimated avg. car width is 1.7m.
- Direct/indirect comparisons: Toyota Corolla is wider than Toyota Corona.
- Distribution: Car width is distributed normally around the average.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1308–1317,
Uppsala, Sweden, 11-16 July 2010.

Usage of all this knowledge allows us to select the correct value of 1.695m and reject other values. Thus we can increase the precision of value extraction by finding and analyzing an entire group of comparable objects.

Second, while it is usually meaningless and impossible to approximate general lexical attribute values like an actor's name, numerical attributes can be estimated *even if they are not explicitly mentioned in the text.*

In general, attribute extraction frameworks usually attempt to discover a single correct value (e.g., capital city of a country) or a set of distinct correct values (e.g., actors of a movie). So there is essentially nothing to do when there is no explicit information present in the text for a given object and an attribute. In contrast, in numerical attribute extraction it is possible to provide an approximation even when no explicit information is present in the text, by using values of comparable objects for which information is provided.

In this paper we present a pattern-based framework that takes advantage of the properties of similar objects to improve extraction precision and allow approximation of requested numerical object properties. Our framework comprises three main stages. First, given an object name we utilize WordNet and pattern-based extraction to find a list of similar objects and their category labels. Second, we utilize a predefined set of lexical patterns in order to extract attribute values of these objects and available comparison/boundary information. Finally, we analyze the obtained information and select or approximate the attribute value for the given (object, attribute) pair.

We performed a thorough evaluation using three different applications: Question Answering (QA), WordNet (WN) enrichment, and comparison with Wikipedia and answers provided by leading search engines. QA evaluation was based on a designed dataset of 1250 questions on size, height, width, weight, and depth, for which we created a gold standard and compared against it automatically[1].

For WN enrichment evaluation, our framework discovered size and weight values for 300 WN physical objects, and the quality of results was evaluated by human judges. For interactive search, we compared our results to information obtained through Wikipedia, Google and Wolfram Alpha. Utilization of information about comparable objects provided a significant boost to numerical attribute extraction quality, and allowed a meaningful approximation of missing attribute values.

Section 2 discusses related work, Section 3 details the algorithmic framework, Section 4 describes the experimental setup, and Section 5 presents our results.

2 Related work

Numerous methods have been developed for extraction of diverse semantic relationships from text. While several studies propose relationship identification methods using distributional analysis of feature vectors (Turney, 2005), the majority of the proposed open-domain relations extraction frameworks utilize lexical patterns connecting a pair of related terms. (Hearst, 1992) manually designed lexico-syntactic patterns for extracting hypernymy relations. (Berland and Charniak, 1999; Girju et al, 2006) proposed a set of patterns for meronymy relations. Davidov and Rappoport (2008a) used pattern clusters to disambiguate nominal compound relations. Extensive frameworks were proposed for iterative discovery of any pre-specified (e.g., (Riloff and Jones, 1999; Chklovski and Pantel, 2004)) and unspecified (e.g., (Banko et al., 2007; Rosenfeld and Feldman, 2007; Davidov and Rappoport, 2008b)) relation types.

The majority of the above methods utilize the following basic strategy. Given (or discovering automatically) a set of patterns or relationship-representing term pairs, these methods mine the web for these patterns and pairs, iteratively obtaining more instances. The proposed strategies generally include some weighting/frequency/context-based algorithms (e.g. (Pantel and Pennacchiotti, 2006)) to reduce noise. Some of the methods are suitable for retrieval of numerical attributes. However, most of them do not exploit the numerical nature of the attribute data.

Our research is related to a sub-domain of question answering (Prager, 2006), since one of the applications of our framework is answering questions on numerical values. The majority of the proposed QA frameworks rely on pattern-based relationship acquisition (Ravichandran and Hovy, 2009). However, most QA studies focus on different types of problems than our paper, including question classification, paraphrasing, etc.

[1]This dataset is available in the authors' websites for the research community.

Several recent studies directly target the acquisition of numerical attributes from the Web and attempt to deal with ambiguity and noise of the retrieved attribute values. (Aramaki et al., 2007) utilize a small set of patterns to extract physical object sizes and use the averages of the obtained values for a noun compound classification task. (Banerjee et al, 2009) developed a method for dealing with quantity consensus queries (QCQs) where there is uncertainty about the answer quantity (e.g. "driving time from Paris to Nice"). They utilize a textual snippet feature and snippet quantity in order to select and rank intervals of the requested values. This approach is particularly useful when it is possible to obtain a substantial amount of a desired attribute values for the requested query. (Moriceau, 2006) proposed a rule-based system which analyzes the variation of the extracted numerical attribute values using information in the textual context of these values.

A significant body of recent research deals with extraction of various data from web tables and lists (e.g., (Cafarella et al., 2008; Crestan and Pantel, 2010)). While in the current research we do not utilize this type of information, incorporation of the numerical data extracted from semi-structured web pages can be extremely beneficial for our framework.

All of the above numerical attribute extraction systems utilize only direct information available in the discovered object-attribute co-occurrences and their contexts. However, as we show, indirect information available for comparable objects can contribute significantly to the selection of the obtained values. Using such indirect information is particularly important when only a modest amount of values can be obtained for the desired object. Also, since the above studies utilize only explicitly available information they were unable to approximate object values in cases where no explicit information was found.

3 The Attribute Mining Framework

Our algorithm is given an object and an attribute. In the WN enrichment scenario, it is also given the object's synset. The algorithm comprises three main stages: (1) mining for similar objects and determination of a class label; (2) mining for attribute values and comparison statements; (3) processing the results.

3.1 Similar objects and class label

To verify and estimate attribute values for the given object we utilize similar objects (co-hyponyms) and the object's class label (hypernym). In the WN enrichment scenario we can easily obtain these, since we get the object's synset as input. However, in Question Answering (QA) scenarios we do not have such information. To obtain it we employ a strategy which uses WordNet along with pattern-based web mining.

Our web mining part follows common pattern-based retrieval practice (Davidov et al., 2007). We utilize Yahoo! Boss API to perform search engine queries. For an object name Obj we query the Web using a small set of pre-defined co-hyponymy patterns like *"as * and/or [Obj]"*[2]. In the WN enrichment scenario, we can add the WN class label to each query in order to restrict results to the desired word sense. In the QA scenario, if we are given the full question and not just the (object, attribute) pair we can add terms appearing in the question and having a strong PMI with the object (this can be estimated using any fixed corpus). However, this is not essential.

We then extract new terms from the retrieved web snippets and use these terms iteratively to retrieve more terms from the Web. For example, when searching for an object 'Toyota', we execute a search engine query *["as * and Toyota"]* and we might retrieve a text snippet containing *"... as Honda and Toyota ... "*. We then extract from this snippet the additional word 'Honda' and use it for iterative retrieval of additional similar terms. We attempt to avoid runaway feedback loop by requiring each newly detected term to co-appear with the original term in at least a single co-hyponymy pattern.

WN class labels are used later for the retrieval of boundary values, and here for expansion of the similar object set. In the WN enrichment scenario, we already have the class label of the object. In the QA scenario, we automatically find class labels as follows. We compute for each WN subtree a coverage value, the number of retrieved terms found in the subtree divided by the number of subtree terms, and select the subtree having the highest coverage. In all scenarios, we add all terms found in this subtree to the retrieved term list. If no WN subtree with significant (> 0.1) coverage is found,

[2] "*" means a search engine wildcard. Square brackets indicate filled slots and are not part of the query.

we retrieve a set of category labels from the Web using hypernymy detection patterns like *"* such as [Obj]"* (Hearst, 1992). If several label candidates were found, we select the most frequent.

Note that we perform this stage only once for each object and do not need to repeat it for different attribute types.

3.2 Querying for values, bounds and comparison data

Now we would like to extract the attribute values for the given object and its similar objects. We will also extract bounds and comparison information in order to verify the extracted values and to approximate the missing ones.

To allow us to extract attribute-specific information, we provided the system with a seed set of extraction patterns for each attribute type. There are three kinds of patterns: value extraction, bounds and comparison patterns. We used up to 10 patterns of each kind. These patterns are the only attribute-specific resource in our framework.

Value extraction. The first pattern group, P_{values}, allows extraction of the attribute values from the Web. All seed patterns of this group contain a measurement unit name, attribute name, and some additional anchoring words, e.g., *'Obj is * [height unit] tall'* or *'Obj width is * [width unit]'*. As in Section 3.1, we execute search engine queries and collect a set of numerical values for each pattern. We extend this group iteratively from the given seed as commonly done in pattern-based acquisition methods. To do this we re-query the Web with the obtained (object, attribute value, attribute name) triplets (e.g., *'[Toyota width 1.695m]'*). We then extract new patterns from the retrieved search engine snippets and re-query the Web with the new patterns to obtain more attribute values.

We provided the framework with unit names and with an appropriate conversion table which allows to convert between different measurement systems and scales. The provided names include common abbreviations like *cm/centimeter*. All value acquisition patterns include unit names, so we know the units of each extracted value. At the end of the value extraction stage, we convert all values to a single unit format for comparison.

Boundary extraction. The second group, $P_{boundary}$, consists of boundary-detection patterns like *'the widest [label] is * [width unit]'*. These patterns incorporate the class labels discovered in the previous stage. They allow us to find maximal and minimal values for the object category defined by labels. If we get several lower bounds and several upper bounds, we select the highest upper bound and the lowest lower bound.

Extraction of comparison information. The third group, $P_{compare}$, consists of comparison patterns. They allow to compare objects directly even when no attribute values are mentioned. This group includes attribute equality patterns such as *'[Object1] has the same width as [Object2]'*, and attribute inequality ones such as *'[Object1] is wider than [Object2]'*. We execute search queries for each of these patterns, and extract a set of ordered term pairs, keeping track of the relationships encoded by the pairs.

We use these pairs to build a directed graph (Widdows and Dorow, 2002; Davidov and Rappoport, 2006) in which nodes are objects (not necessarily with assigned values) and edges correspond to extracted co-appearances of objects inside the comparison patterns. The directions of edges are determined by the comparison sign. If two objects co-appear inside an equality pattern we put a bidirectional edge between them.

3.3 Processing the collected data

As a result of the information collection stage, for each object and attribute type we get:

- A set of attribute values for the requested object.
- A set of objects similar or comparable to the requested object, some of them annotated with one or many attribute values.
- Upper and lowed bounds on attribute values for the given object category.
- A comparison graph connecting some of the retrieved objects by comparison edges.

Obviously, some of these components may be missing or noisy. Now we combine these information sources to select a single attribute value for the requested object or to approximate this value. First we apply bounds, removing out-of-range values, then we use comparisons to remove inconsistent comparisons. Finally we examine the remaining values and the comparison graph.

Processing bounds. First we verify that indeed most ($\geq 50\%$) of the retrieved values fit the retrieved bounds. If the lower and/or upper bound

contradicts more than half of the data, we reject the bound. Otherwise we remove all values which do not satisfy one or both of the accepted bounds. If no bounds are found or if we disable the bound retrieval (see Section 4.1), we assign the maximal and minimal observed values as bounds.

Since our goal is to obtain a value for the single requested object, if at the end of this stage we remain with a single value, no further processing is needed. However, if we obtain a set of values or no values at all, we have to utilize comparison data to select one of the retrieved values or to approximate the value in case we do not have an exact answer.

Processing comparisons. First we simplify the comparison graph. We drop all graph components that are not connected (when viewing the graph as undirected) to the desired object.

Now we refine the graph. Note that each graph node may have a single value, many assigned values, or no assigned values. We define *assigned nodes* as nodes that have at least one value. For each directed edge $E(A \rightarrow B)$, if both A and B are assigned nodes, we check if $Avg(A) \leq Avg(B)$[3]. If the average values violate the equation, we gradually remove up to half of the highest values for A and up to half of the lowest values for B till the equation is satisfied. If this cannot be done, we drop the edge. We repeat this process until every edge that connects two assigned nodes satisfies the inequality.

Selecting an exact attribute value. The goal now is to select an attribute value for the given object. During the first stage it is possible that we directly extract from the text a set of values for the requested object. The bounds processing step rejects some of these values, and the comparisons step may reject some more. If we still have several values remaining, we choose the most frequent value based on the number of web snippets retrieved during the value acquisition stage. If there are several values with the same frequency we select the median of these values.

Approximating the attribute value. In the case when we do not have any values remaining after the bounds processing step, the object node will remain unassigned after construction of the comparison graph, and we would like to estimate its value. Here we present an algorithm which allows us to set the values of all unassigned nodes, including the node of the requested object.

In the algorithm below we treat all node groups connected by bidirectional (equality) edges as a same-value group, i.e., if a value is assigned to one node in the group, the same value is immediately assigned to the rest of the nodes in the same group.

We start with some preprocessing. We create dummy lower and upper bound nodes L and U with corresponding upper/lower bound values obtained during the previous stage. These dummy nodes will be used when we encounter a graph which ends with one or more nodes with no available numerical information. We then connect them to the graph as follows: (1) if A has no incoming edges, we add an edge $L \rightarrow A$; (2) if A has no outgoing edges, we add an edge $A \rightarrow U$.

We define a *legal unassigned path* as a directed path $A_0 \rightarrow A_1 \rightarrow \ldots \rightarrow A_n \rightarrow A_{n+1}$ where A_0 and A_{n+1} are assigned satisfying $Avg(A_0) \leq Avg(A_{n+1})$ and $A_1 \ldots A_n$ are unassigned. We would like to use dummy bound nodes only in cases when no other information is available. Hence we consider paths $L \rightarrow \ldots \rightarrow U$ connecting both bounds are illegal. First we assign values for all unassigned nodes that belong to a single legal unassigned path, using a simple linear combination:
$Val(A_i)_{i \in (1 \ldots n)} =$

$$\frac{n+1-i}{n+1} Avg(A_0) + \frac{i}{n+1} Avg(A_{n+1})$$

Then, for all unassigned nodes that belong to multiple legal unassigned paths, we compute node value as above for each path separately and assign to the node the average of the computed values.

Finally we assign the average of all extracted values within bounds to all the remaining unassigned nodes. Note that if we have no comparison information and no value information for the requested object, the requested object will receive the average of the extracted values of the whole set of the retrieved comparable objects and the comparison step will be essentially empty.

4 Experimental Setup

We performed automated question answering (QA) evaluation, human-based WN enrichment evaluation, and human-based comparison of our results to data available through Wikipedia and to the top results of leading search engines.

[3] Avg. is of values of an object, without similar objects.

4.1 Experimental conditions

In order to test the main system components, we ran our framework under five different conditions:

- **FULL:** All system components were used.
- **DIRECT:** Only direct pattern-based acquisition of attribute values (Section 3.2, value extraction) for the given object was used, as done in most general-purpose attribute acquisition systems. If several values were extracted, the most common value was used as an answer.
- **NOCB:** No boundary and no comparison data were collected and processed ($P_{compare}$ and P_{bounds} were empty). We only collected and processed a set of values for the similar objects.
- **NOB:** As in **FULL** but no boundary data was collected and processed (P_{bounds} was empty).
- **NOC:** As in **FULL** but no comparison data was collected and processed ($P_{compare}$ was empty).

4.2 Automated QA Evaluation

We created two QA datasets, Web and TREC based.

Web-based QA dataset. We created QA datasets for size, height, width, weight, and depth attributes. For each attribute we extracted from the Web 250 questions in the following way. First, we collected several thousand questions, querying for the following patterns: "How long/tall/wide/heavy/deep/high is","What is the size/width/height/depth/weight of". Then we manually filtered out non-questions and heavily context-specific questions, e.g., *"what is the width of the triangle"*. Next, we retained only a single question for each entity by removing duplicates.

For each of the extracted questions we manually assigned a gold standard answer using trusted resources including books and reliable Web data. For some questions, the exact answer is the only possible one (e.g., the height of a person), while for others it is only the center of a distribution (e.g., the weight of a coffee cup). Questions with no trusted and exact answers were eliminated. From the remaining questions we randomly selected 250 questions for each attribute.

TREC-based QA dataset. As a small complementary dataset we used relevant questions from the TREC Question Answering Track 1999-2007. From 4355 questions found in this set we collected 55 (17 size, 2 weight, 3 width, 3 depth and 30 height) questions.

Examples. Some example questions from our datasets are (correct answers are in parentheses): How tall is Michelle Obama? (180cm); How tall is the tallest penguin? (122cm); What is the height of a tennis net? (92cm); What is the depth of the Nile river? (1000cm = 10 meters); How heavy is a cup of coffee? (360gr); How heavy is a giraffe? (1360000gr = 1360kg); What is the width of a DNA molecule? (2e-7cm); What is the width of a cow? (65cm).

Evaluation protocol. Evaluation against the datasets was done automatically. For each question and each condition our framework returned a numerical value marked as either an exact answer or as an approximation. In cases where no data was found for an approximation (no similar objects with values were found), our framework returned no answer.

We computed precision[4], comparing results to the gold standard. Approximate answers are considered to be correct if the approximation is within 10% of the gold standard value. While a choice of 10% may be too strict for some applications and too generous for others, it still allows to estimate the quality of our framework.

4.3 WN enrichment evaluation

We manually selected 300 WN entities from about 1000 randomly selected objects below the object tree in WN, by filtering out entities that clearly do not possess any of the addressed numerical attributes.

Evaluation was done using human subjects. It is difficult to do an automated evaluation, since the nature of the data is different from that of the QA dataset. Most of the questions asked over the Web target named entities like specific car brands, places and actors. There is usually little or no variability in attribute values of such objects, and the major source of extraction errors is name ambiguity of the requested objects.

WordNet physical objects, in contrast, are much less specific and their attributes such as size and

[4]Due to the nature of the task recall/f-score measures are redundant here

weight rarely have a single correct value, but usually possess an acceptable numerical range. For example, the majority of the selected objects like 'apple' are too general to assign an exact size. Also, it is unclear how to define acceptable values and an approximation range. Crudeness of desired approximation depends both on potential applications and on object type. Some objects show much greater variability in size (and hence a greater range of acceptable approximations) than others. This property of the dataset makes it difficult to provide a meaningful gold standard for the evaluation. Hence in order to estimate the quality of our results we turn to an evaluation based on human judges.

In this evaluation we use only *approximate* retrieved values, keeping out the small amount of returned *exact* values[5].

We have mixed (Object, Attribute name, Attribute value) triplets obtained through each of the conditions, and asked human subjects to assign these to one of the following categories:

- The attribute value is reasonable for the given object.
- The value is a very crude approximation of the given object attribute.
- The value is incorrect or clearly misleading.
- The object is not familiar enough to me so I cannot answer the question.

Each evaluator was provided with a random sample of 40 triplets. In addition we mixed in 5 manually created clearly correct triplets and 5 clearly incorrect ones. We used five subjects, and the agreement (inter-annotator Kappa) on shared evaluated triplets was 0.72.

4.4 Comparisons to search engine output

Recently there has been a significant improvement both in the quality of search engine results and in the creation of manual well-organized and annotated databases such as Wikipedia.

Google and Yahoo! queries frequently provide attribute values in the top snippets or in search result web pages. Many Wikipedia articles include infoboxes with well-organized attribute values. Recently, the Wolfram Alpha computational knowledge engine presented the computation of attribute values from a given query text.

[5]So our results are in fact higher than shown.

Hence it is important to test how well our framework can complement the manual extraction of attributes from resources such as Wikipedia and top Google snippets. In order to test this, we randomly selected 100 object-attribute pairs from our Web QA and WordNet datasets and used human subjects to test the following:

1. **Go1**: Querying Google for *[object-name attribute-name]* gives in some of the first three snippets a correct value or a good approximation value[6] for this pair.
2. **Go2**: Querying Google for *[object-name attribute-name]* and following the first three links gives a correct value or a good approximation value.
3. **Wi**: There is a Wikipedia page for the given object and it contains an appropriate attribute value or an approximation in an infobox.
4. **Wf**: A Wolfram Alpha query for *[object-name attribute-name]* retrieves a correct value or a good approximation value

5 Results

5.1 QA results

We applied our framework to the above QA datasets. Table 1 shows the precision and the percentage of approximations and exact answers.

Looking at %Exact+%Approx, we can see that for all datasets only 1-9% of the questions remain unanswered, while correct exact answers are found for 65%/87% of the questions for Web/TREC (% Exact and Prec(Exact) in the table). Thus approximation allows us to answer 13-24% of the requested values which are either simply missing from the retrieved text or cannot be detected using the current pattern-based framework. Comparing performance of FULL to DIRECT, we see that our framework not only allows an approximation when no exact answer can be found, but also significantly increases the precision of exact answers using the comparison and the boundary information. It is also apparent that both boundary and comparison features are needed to achieve good performance and that using both of them achieves substantially better results than each of them separately.

[6]As defi ned in the human subject questionnaire.

	FULL	DIRECT	NOCB	NOB	NOC
Web QA					
Size					
%Exact	80	82	82	82	80
Prec(Exact)	76	40	40	54	65
%Approx	16	-	14	14	16
Prec(Appr)	64	-	34	53	46
Height					
%Exact	79	84	84	84	79
Prec(Exact)	86	56	56	69	70
%Approx	16	-	11	11	16
Prec(Appr)	72	-	25	65	53
Width					
%Exact	74	76	76	76	74
Prec(Exact)	86	45	45	60	72
%Approx	17	-	15	15	17
Prec(Appr)	75	-	26	63	55
Weight					
%Exact	71	73	73	73	71
Prec(Exact)	82	57	57	64	70
Prec(Appr)	24	-	22	22	24
%Approx	61	-	39	51	46
Depth					
%Exact	82	82	82	82	82
Prec(Exact)	89	60	60	71	78
%Approx	19	-	19	19	19
Prec(Appr)	92	-	58	76	63
Total average					
%Exact	77	79	79	79	77
Prec(Exact)	84	52	52	64	71
%Approx	18	-	16	16	19
Prec(Appr)	72	-	36	62	53
TREC QA					
%Exact	87	90	90	90	87
Prec(Exact)	**100**	62	62	84	76
%Approx	13	-	9	9	13
Prec(Appr)	**57**	-	20	40	57

Table 1: Precision and amount of exact and approximate answers for QA datasets.

Comparing results for different question types we can see substantial performance differences between the attribute types. Thus depth shows much better overall results than width. This is likely due to a lesser difficulty of depth questions or to a more exact nature of available depth information compared to width or size.

5.2 WN enrichment

As shown in Table 2, for the majority of examined WN objects, the algorithm returned an approximate value, and only for 13-15% of the objects (vs. 70-80% in QA data) the algorithm could retrieve exact answers.

Note that the common pattern-based acquisition framework, presented as the DIRECT condition, could only extract attribute values for 15% of the objects since it does not allow approximations and may only extract values from the text where they explicitly appear.

	FULL	DIRECT	NOCB	NOB	NOC
Size					
%Exact	15.3	18.0	18.0	18.0	15.3
%Approx	80.3	-	38.2	20.0	23.6
Weight					
%Exact	11.8	12.5	12.5	12.5	11.8
%Approx	71.7	-	38.2	20.0	23.6

Table 2: Percentage of exact and approximate values for the WordNet enrichment dataset.

	FULL	NOCB	NOB	NOC
Size				
%Correct	73	21	49	28
%Crude	15	54	31	49
%Incorrect	8	21	16	19
Weight				
%Correct	64	24	46	38
%Crude	24	45	30	41
%Incorrect	6	25	18	15

Table 3: Human evaluation of approximations for the WN enrichment dataset (the percentages are averaged over the human subjects).

Table 3 shows human evaluation results. We see that the majority of approximate values were clearly accepted by human subjects, and only 6-8% were found to be incorrect. We also observe that both boundary and comparison data significantly improve the approximation results. Note that DIRECT is missing from this table since no approximations are possible in this condition.

Some examples for WN objects and approximate values discovered by the algorithm are: Sandfish, 15gr; skull, 1100gr; pilot, 80.25kg. The latter value is amusing due to the high variability of the value. However, even this value is valuable, as a sanity check measure for automated inference systems and for various NLP tasks (e.g., 'pilot jacket' likely refers to a jacket used by pilots and not vice versa).

5.3 Comparison with search engines and Wikipedia

Table 4 shows results for the above datasets in comparison to the proportion of correct results and the approximations returned by our framework under the FULL condition (correct exact values and approximations are taken together).

We can see that our framework, due to its approximation capability, currently shows significantly greater coverage than manual extraction of data from Wikipedia infoboxes or from the first

	FULL	Go1	Go2	Wi	Wf
Web QA	83	32	40	15	21
WordNet	87	24	27	18	5

Table 4: Comparison of our attribute extraction framework to manual extraction using Wikipedia and search engines.

search engine results.

6 Conclusion

We presented a novel framework which allows an automated extraction and approximation of numerical attributes from the Web, even when no explicit attribute values can be found in the text for the given object. Our framework retrieves similarity, boundary and comparison information for objects similar to the desired object, and combines this information to approximate the desired attribute.

While in this study we explored only several specific numerical attributes like size and weight, our framework can be easily augmented to work with any other consistent and comparable attribute type. The only change required for incorporation of a new attribute type is the development of attribute-specific $P_{boundary}$, P_{values}, and $P_{compare}$ pattern groups; the rest of the system remains unchanged.

In our evaluation we showed that our framework achieves good results and significantly outperforms the baseline commonly used for general lexical attribute retrieval[7].

While there is a growing justification to rely on extensive manually created resources such as Wikipedia, we have shown that in our case automated numerical attribute acquisition could be a preferable option and provides excellent coverage in comparison to handcrafted resources or manual examination of the leading search engine results. Hence a promising direction would be to use our approach in combination with Wikipedia data and with additional manually created attribute rich sources such as Web tables, to achieve the best possible performance and coverage.

We would also like to explore the incorporation of approximate discovered numerical attribute data into existing NLP tasks such as noun compound classification and textual entailment.

[7]It should be noted, however, that in our DIRECT baseline we used a basic pattern-based retrieval strategy; more sophisticated strategies for value selection might bring better results.

References

Eiji Aramaki, Takeshi Imai, Kengo Miyo and Kazuhiko Ohe. 2007 UTH: SVM-based Semantic Relation Classifi cation using Physical Sizes. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*.

Somnath Banerjee, Soumen Chakrabarti and Ganesh Ramakrishnan. 2009. Learning to Rank for Quantity Consensus Queries. *SIGIR '09*.

Michele Banko, Michael J Cafarella , Stephen Soderland, Matt Broadhead and Oren Etzioni. 2007. Open information extraction from the Web. *IJCAI '07*.

Matthew Berland, Eugene Charniak, 1999. Finding parts in very large corpora. *ACL '99*.

Michael Cafarella, Alon Halevy, Yang Zhang, Daisy Zhe Wang and Eugene Wu. 2008. WebTables: Exploring the Power of Tables on the Web. *VLDB '08*.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: mining the Web for fi ne-grained semantic verb relations. *EMNLP '04*.

Eric Crestan and Patrick Pantel. 2010. Web-Scale Knowledge Extraction from Semi-Structured Tables. *WWW '10*.

Dmitry Davidov and Ari Rappoport. 2006. Effi cient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. *ACL-Coling '06.*

Dmitry Davidov, Ari Rappoport and Moshe Koppel. 2007. Fully unsupervised discovery of concept-specifi c relationships by web mining. *ACL '07*.

Dmitry Davidov and Ari Rappoport. 2008a. Classifi - cation of Semantic Relationships between Nominals Using Pattern Clusters. *ACL '08.*

Dmitry Davidov and Ari Rappoport. 2008b. Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions. *ACL '08.*

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1).

Marty Hearst, 1992. Automatic acquisition of hyponyms from large text corpora. *COLING '92*.

Veronique Moriceau, 2006. Numerical Data Integration for Cooperative Question-Answering. *EACL - KRAQ06 '06*.

John Prager, 2006. Open-domain question-answering. *In Foundations and Trends in Information Retrieval,vol. 1, pp 91-231*.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. *COLING-ACL '06*.

Deepak Ravichandran and Eduard Hovy. 2002 Learning Surface Text Patterns for a Question Answering System. *ACL '02*.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *AAAI '99*.

Benjamin Rosenfeld and Ronen Feldman. 2007. Clustering for unsupervised relation identification. *CIKM '07*.

Peter Turney, 2005. Measuring semantic similarity by latent relational analysis, *IJCAI '05*.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised Lexical acquisition. *COLING '02*.

Learning Word-Class Lattices for Definition and Hypernym Extraction

Roberto Navigli and **Paola Velardi**
Dipartimento di Informatica
Sapienza Università di Roma
{navigli,velardi}@di.uniroma1.it

Abstract

Definition extraction is the task of automatically identifying definitional sentences within texts. The task has proven useful in many research areas including ontology learning, relation extraction and question answering. However, current approaches – mostly focused on lexico-syntactic patterns – suffer from both low recall and precision, as definitional sentences occur in highly variable syntactic structures. In this paper, we propose Word-Class Lattices (WCLs), a generalization of word lattices that we use to model textual definitions. Lattices are learned from a dataset of definitions from Wikipedia. Our method is applied to the task of definition and hypernym extraction and compares favorably to other pattern generalization methods proposed in the literature.

1 Introduction

Textual definitions constitute a fundamental source to look up when the meaning of a term is sought. Definitions are usually collected in dictionaries and domain glossaries for consultation purposes. However, manually constructing and updating glossaries requires the cooperative effort of a team of domain experts. Further, in the presence of new words or usages, and – even worse – new domains, such resources are of no help. Nonetheless, terms are attested in texts and some (usually few) of the sentences in which a term occurs are typically definitional, that is they provide a formal explanation for the term of interest. While it is not feasible to manually search texts for definitions, this task can be automatized by means of Machine Learning (ML) and Natural Language Processing (NLP) techniques.

Automatic definition extraction is useful not only in the construction of glossaries, but also in many other NLP tasks. In ontology learning, definitions are used to create and enrich concepts with textual information (Gangemi et al., 2003), and extract taxonomic and non-taxonomic relations (Snow et al., 2004; Navigli and Velardi, 2006; Navigli, 2009a). Definitions are also harvested in Question Answering to deal with "what is" questions (Cui et al., 2007; Saggion, 2004). In eLearning, they are used to help students assimilate knowledge (Westerhout and Monachesi, 2007), etc.

Much of the current literature focuses on the use of lexico-syntactic patterns, inspired by Hearst's (1992) seminal work. However, these methods suffer both from low recall and precision, as definitional sentences occur in highly variable syntactic structures, and because the most frequent definitional pattern – X is a Y – is inherently very noisy.

In this paper we propose a generalized form of word lattices, called Word-Class Lattices (WCLs), as an alternative to lexico-syntactic pattern learning. A lattice is a directed acyclic graph (DAG), a subclass of non-deterministic finite state automata (NFA). The lattice structure has the purpose of preserving the salient differences among distinct sequences, while eliminating redundant information. In computational linguistics, lattices have been used to model in a compact way many sequences of symbols, each representing an alternative hypothesis. Lattice-based methods differ in the types of nodes (words, phonemes, concepts), the interpretation of links (representing either a sequential or hierarchical ordering between nodes), their means of creation, and the scoring method used to extract the best consensus output from the lattice (Schroeder et al., 2009). In speech processing, phoneme or word lattices (Campbell et al., 2007; Mathias and Byrne, 2006; Collins et al., 2004) are used as an interface between speech recognition and understanding. Lat-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1318–1327,
Uppsala, Sweden, 11-16 July 2010.

tices are adopted also in Chinese word segmentation (Jiang et al., 2008), decompounding in German (Dyer, 2009), and to represent classes of translation models in machine translation (Dyer et al., 2008; Schroeder et al., 2009). In more complex text processing tasks, such as information retrieval, information extraction and summarization, the use of word lattices has been postulated but is considered unrealistic because of the dimension of the hypothesis space.

To reduce this problem, concept lattices have been proposed (Carpineto and Romano, 2005; Klein, 2008; Zhong et al., 2008). Here links represent hierarchical relations, rather than the sequential order of symbols like in word/phoneme lattices, and nodes are clusters of salient words aggregated using synonymy, similarity, or subtrees of a thesaurus. However, salient word selection and aggregation is non-obvious and furthermore it falls into word sense disambiguation, a notoriously AI-hard problem (Navigli, 2009b).

In definition extraction, the variability of patterns is higher than for "traditional" applications of lattices, such as translation and speech, however not as high as in unconstrained sentences. The methodology that we propose to align patterns is based on the use of star (wildcard *) characters to facilitate sentence clustering. Each cluster of sentences is then generalized to a lattice of word classes (each class being either a frequent word or a part of speech). A key feature of our approach is its inherent ability to both identify definitions and extract hypernyms. The method is tested on an annotated corpus of Wikipedia sentences and a large Web corpus, in order to demonstrate the independence of the method from the annotated dataset. WCLs are shown to generalize over lexico-syntactic patterns, and outperform well-known approaches to definition and hypernym extraction.

The paper is organized as follows: Section 2 discusses related work, WCLs are introduced in Section 3 and illustrated by means of an example in Section 4, experiments are presented in Section 5. We conclude the paper in Section 6.

2 Related Work

Definition Extraction. A great deal of work is concerned with definition extraction in several languages (Klavans and Muresan, 2001; Storrer and Wellinghoff, 2006; Gaudio and Branco, 2007; Iftene et al., 2007; Westerhout and Monachesi, 2007; Przepiórkowski et al., 2007; Degórski et al., 2008). The majority of these approaches use symbolic methods that depend on lexico-syntactic patterns or features, which are manually crafted or semi-automatically learned (Zhang and Jiang, 2009; Hovy et al., 2003; Fahmi and Bouma, 2006; Westerhout, 2009). Patterns are either very simple sequences of words (e.g. "refers to", "is defined as", "is a") or more complex sequences of words, parts of speech and chunks. A fully automated method is instead proposed by Borg et al. (2009): they use genetic programming to learn simple features to distinguish between definitions and non-definitions, and then they apply a genetic algorithm to learn individual weights of features. However, rules are learned for only one category of patterns, namely "is" patterns. As we already remarked, most methods suffer from both low recall and precision, because definitional sentences occur in highly variable and potentially noisy syntactic structures. Higher performance (around 60-70% F_1-measure) is obtained only for specific domains (e.g., an ICT corpus) and patterns (Borg et al., 2009).

Only few papers try to cope with the generality of patterns and domains in real-world corpora (like the Web). In the GlossExtractor web-based system (Velardi et al., 2008), to improve precision while keeping pattern generality, candidates are pruned using more refined stylistic patterns and lexical filters. Cui et al. (2007) propose the use of probabilistic lexico-semantic patterns, called *soft patterns*, for definitional question answering in the TREC contest[1]. The authors describe two soft matching models: one is based on an n-gram language model (with the Expectation Maximization algorithm used to estimate the model parameter), the other on Profile Hidden Markov Models (PHMM). Soft patterns generalize over lexico-syntactic "hard" patterns in that they allow a partial matching by calculating a generative degree of match probability between the test instance and the set of training instances. Thanks to its generalization power, this method is the most closely related to our work, however the task of definitional question answering to which it is applied is slightly different from that of definition extraction, so a direct performance comparison is not possi-

[1]Text REtrieval Conferences: `http://trec.nist.gov`

ble[2]. In fact, the TREC evaluation datasets cannot be considered true definitions, but rather text fragments providing some relevant fact about a target term. For example, sentences like: "Bollywood is a Bombay-based film industry" and "700 or more films produced by India with 200 or more from Bollywood" are both "vital" answers for the question "Bollywood", according to TREC classification, but the second sentence is not a definition.

Hypernym Extraction. The literature on hypernym extraction offers a higher variability of methods, from simple lexical patterns (Hearst, 1992; Oakes, 2005) to statistical and machine learning techniques (Agirre et al., 2000; Caraballo, 1999; Dolan et al., 1993; Sanfilippo and Poznański, 1992; Ritter et al., 2009). One of the highest-coverage methods is proposed by Snow et al. (2004). They first search sentences that contain two terms which are known to be in a taxonomic relation (term pairs are taken from WordNet (Miller et al., 1990)); then they parse the sentences, and automatically extract patterns from the parse trees. Finally, they train a hypernym classifer based on these features. Lexico-syntactic patterns are generated for each sentence relating a term to its hypernym, and a dependency parser is used to represent them.

3 Word-Class Lattices

3.1 Preliminaries

Notion of definition. In our work, we rely on a formal notion of textual definition. Specifically, given a definition, e.g.: "In computer science, a closure is a first-class function with free variables that are bound in the lexical environment", we assume that it contains the following fields (Storrer and Wellinghoff, 2006):

- The DEFINIENDUM field (DF): this part of the definition includes the *definiendum* (that is, the word being defined) and its modifiers (e.g., "In computer science, a closure");
- The DEFINITOR field (VF): it includes the verb phrase used to introduce the definition (e.g., "is");
- The DEFINIENS field (GF): it includes the genus phrase (usually including the hypernym, e.g., "a first-class function");
- The REST field (RF): it includes additional clauses that further specify the *differentia* of the definiendum with respect to its genus (e.g., "with free variables that are bound in the lexical environment").

Further examples of definitional sentences annotated with the above fields are shown in Table 1. For each sentence, the definiendum (that is, the word being defined) and its hypernym are marked in bold and italic, respectively. Given the lexico-syntactic nature of the definition extraction models we experiment with, training and test sentences are part-of-speech tagged with the TreeTagger system, a part-of-speech tagger available for many languages (Schmid, 1995).

Word Classes and Generalized Sentences. We now introduce our notion of word class, on which our learning model is based. Let $\mathcal{T}$ be the set of training sentences, manually bracketed with the DF, VF, GF and RF fields. We first determine the set F of words in $\mathcal{T}$ whose frequency is above a threshold θ (e.g., *the*, *a*, *is*, *of*, *refer*, etc.). In our training sentences, we replace the term being defined with ⟨TARGET⟩, thus this frequent token is also included in F.

We use the set of frequent words F to generalize words to "word classes". We define a word class as either a word itself or its part of speech. Given a sentence $s = w_1, w_2, \ldots, w_{|s|}$, where w_i is the i-th word of s, we generalize its words w_i to word classes ω_i as follows:

$$\omega_i = \begin{cases} w_i & \text{if } w_i \in F \\ POS(w_i) & \text{otherwise} \end{cases}$$

that is, a word w_i is left unchanged if it occurs frequently in the training corpus (i.e., $w_i \in F$) or is transformed to its part of speech ($POS(w_i)$) otherwise. As a result, we obtain a generalized sentence $s' = \omega_1, \omega_2, \ldots, \omega_{|s|}$. For instance, given the first sentence in Table 1, we obtain the corresponding generalized sentence: "In NN, a ⟨TARGET⟩ is a JJ NN", where NN and JJ indicate the noun and adjective classes, respectively.

3.2 Algorithm

We now describe our learning algorithm based on Word-Class Lattices. The algorithm consists of three steps:

[2] In the paper, a 55% recall and 34% precision is achieved with the best experiment on TREC-13 data. Furthermore, the classifier of Cui et al. (2007) is based on soft patterns but also on a bag-of-word relevance heuristic. However, the relative influence of the two methods on the final performance is not discussed.

[In arts, a **chiaroscuro**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a monochrome *picture*]$_{\text{GF}}$.
[In mathematics, a **graph**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a *data structure*]$_{\text{GF}}$ [that consists of ...]$_{\text{REST}}$.
[In computer science, a **pixel**]$_{\text{DF}}$ [is]$_{\text{VF}}$ [a *dot*]$_{\text{GF}}$ [that is part of a computer image]$_{\text{REST}}$.

Table 1: Example definitions (defined terms are marked in bold face, their hypernyms in italic).

- **Star patterns:** each sentence in the training set is pre-processed and generalized to a star pattern. For instance, "In arts, a chiaroscuro is a monochrome picture" is transformed to "In *, a ⟨TARGET⟩ is a *" (Section 3.2.1);
- **Sentence clustering:** the training sentences are then clustered based on the star patterns to which they belong (Section 3.2.2);
- **Word-Class Lattice construction:** for each sentence cluster, a WCL is created by means of a greedy alignment algorithm (Section 3.2.3).

We present two variants of our WCL model, dealing either globally with the entire sentence or separately with its definition fields (Section 3.2.4). The WCL models can then be used to classify any input sentence of interest (Section 3.2.5).

3.2.1 Star Patterns

Let $\mathcal{T}$ be the set of training sentences. In this step, we associate a star pattern $\sigma(s)$ with each sentence $s \in \mathcal{T}$. To do so, let $s \in \mathcal{T}$ be a sentence such that $s = w_1, w_2, \ldots, w_{|s|}$, where w_i is its i-th word. Given the set F of most frequent words in $\mathcal{T}$ (cf. Section 3.1), the star pattern $\sigma(s)$ associated with s is obtained by replacing with * all the words $w_i \notin F$, that is all the tokens that are non-frequent words. For instance, given the sentence "In arts, a chiaroscuro is a monochrome picture", the corresponding star pattern is "In *, a ⟨TARGET⟩ is a *", where ⟨TARGET⟩ is the defined term.

Note that, here and in what follows, we discard the sentence fragments tagged with the REST field, which is used only to delimit the core part of definitional sentences.

3.2.2 Sentence Clustering

In the second step, we cluster the sentences in our training set $\mathcal{T}$ based on their star patterns. Formally, let $\Sigma = (\sigma_1, \ldots, \sigma_m)$ be the set of star patterns associated with the sentences in $\mathcal{T}$. We create a clustering $\mathcal{C} = (C_1, \ldots, C_m)$ such that $C_i = \{s \in \mathcal{T} : \sigma(s) = \sigma_i\}$, that is C_i contains all the sentences whose star pattern is σ_i.

As an example, assume σ_3 = "In *, a ⟨TARGET⟩ is a *". The sentences reported in Table 1 are all grouped into cluster C_3. We note that each cluster C_i contains sentences whose degree of variability is generally much lower than for any pair of sentences in $\mathcal{T}$ belonging to two different clusters.

3.2.3 Word-Class Lattice Construction

Finally, the third step consists of the construction of a Word-Class Lattice for each sentence cluster. Given such a cluster $C_i \in \mathcal{C}$, we apply a greedy algorithm that iteratively constructs the WCL.

Let $C_i = \{s_1, s_2, \ldots, s_{|C_i|}\}$ and consider its first sentence $s_1 = w_1^1, w_2^1, \ldots, w_{|s_1|}^1$ (w_i^j denotes the i-th token of the j-th sentence). We first produce the corresponding generalized sentence $s'_1 = \omega_1^1, \omega_2^1, \ldots, \omega_{|s_1|}^1$ (cf. Section 3.1). We then create a directed graph $G = (V, E)$ such that $V = \{\omega_1^1, \ldots, \omega_{|s_1|}^1\}$ and $E = \{(\omega_1^1, \omega_2^1), (\omega_2^1, \omega_3^1), \ldots, (\omega_{|s_1|-1}^1, \omega_{|s_1|}^1)\}$. Next, for the subsequent sentences in C_i, that is, for each $j = 2, \ldots, |C_i|$, we determine the alignment between the sentence s_j and each sentence $s_k \in C_i$ such that $k < j$ based on the following dynamic programming formulation (Cormen et al., 1990, pp. 314–319):

$$M_{a,b} = \max \{M_{a-1,b-1} + S_{a,b}, M_{a,b-1}, M_{a-1,b}\}$$

where $a \in \{1, \ldots, |s_k|\}$ and $b \in \{1, \ldots, |s_j|\}$, $S_{a,b}$ is a score of the matching between the a-th token of s_k and the b-th token of s_j, and $M_{0,0}$, $M_{0,b}$ and $M_{a,0}$ are initially set to 0 for all a and b.

The matching score $S_{a,b}$ is calculated on the generalized sentences s'_k of s_k and s'_j of s_j as follows:

$$S_{a,b} = \begin{cases} 1 & \text{if } \omega_a^k = \omega_b^j \\ 0 & \text{otherwise} \end{cases}$$

where ω_a^k and ω_b^j are the a-th and b-th word classes of s'_k and s'_j, respectively. In other words, the matching score equals 1 if the a-th and the b-th tokens of the two original sentences have the same word class.

Finally, the alignment score between s_k and s_j is given by $M_{|s_k|,|s_j|}$, which calculates the mini-

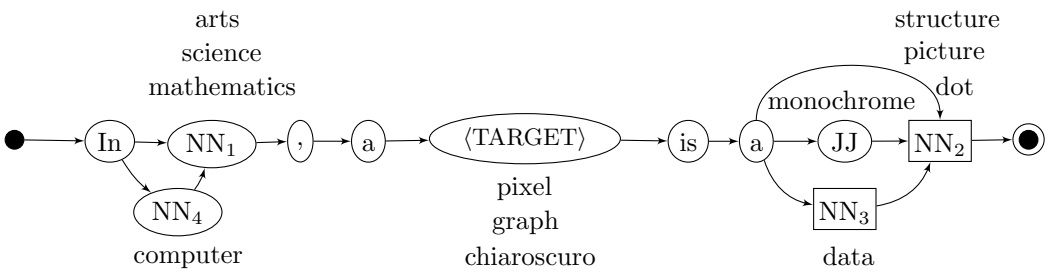


Figure 1: The Word-Class Lattice for the sentences in Table 1. The support of each word class is reported beside the corresponding node.

mal number of misalignments between the two token sequences. We repeat this calculation for each sentence s_k ($k = 1, \ldots, j-1$) and choose the one that maximizes its alignment score with s_j. We then use the best alignment to add s_j to the graph G. Such alignment is obtained by means of backtracking from $M_{|s_k|,|s_j|}$ to $M_{0,0}$. We add to the set of vertices V the tokens of the generalized sentence s'_j for which there is no alignment to s'_k and we add to E the edges (ω^j_1, ω^j_2), $\ldots, (\omega^j_{|s_j|-1}, \omega^j_{|s_j|})$. Furthermore, in the final lattice, nodes associated with the hypernym words in the learning sentences are marked as hypernyms in order to be able to determine the hypernym of a test sentence at classification time.

3.2.4 Variants of the WCL Model

So far, we have assumed that our WCL model learns lattices from the training sentences in their entirety (we call this model WCL-1). We now propose a second model that learns separate WCLs for each field of the definition, namely: the DEFINIENDUM (DF), DEFINITOR (VF) and DEFINIENS (GF) fields (see Section 3.1). We refer to this latter model as WCL-3. Rather than applying the WCL algorithm to the entire sentence, the very same method is applied to the sentence fragments tagged with one of the three definition fields. The reason for introducing the WCL-3 model is that, while definitional patterns are highly variable, DF, VF and GF individually exhibit a lower variability, thus WCL-3 should improve the generalization power.

3.2.5 Classification

Once the learning process is over, a set of WCLs is produced. Given a test sentence s, the classification phase for the WCL-1 model consists of determining whether it exists a lattice that matches s. In the case of WCL-3, we consider any combination of DEFINIENDUM, DEFINITOR and DEFINIENS lattices. While WCL-1 is applied as a yes-no classifier as there is a single WCL that can possibly match the input sentence, WCL-3 selects, if any, the combination of the three WCLs that best fits the sentence. In fact, choosing the most appropriate combination of lattices impacts the performance of hypernym extraction. The best combination of WCLs is selected by maximizing the following confidence score:

$$score(s, l_{\text{DF}}, l_{\text{VF}}, l_{\text{GF}}) = \textit{coverage} \cdot log(\textit{support})$$

where s is the candidate sentence, l_{DF}, l_{VF} and l_{GF} are three lattices one for each definition field, *coverage* is the fraction of words of the input sentence covered by the three lattices, and *support* is the sum of the number of sentences in the star patterns corresponding to the three lattices.

Finally, when a sentence is classified as a definition, its hypernym is extracted by selecting the words in the input sentence that are marked as "hypernyms" in the WCL-1 lattice (or in the WCL-3 GF lattice).

4 Example

As an example, consider the definitions in Table 1. As illustrated in Section 3.2.2, their star pattern is "In *, a ⟨TARGET⟩ is a *". The corresponding WCL is built as follows: the first part-of-speech tagged sentence, "In/IN arts/NN , a/DT ⟨TARGET⟩/NN is/VBZ a/DT monochrome/JJ *picture*/NN", is considered. The corresponding generalized sentence is "In NN , a ⟨TARGET⟩ is a JJ NN". The initially empty graph is thus populated with one node for each word class and one edge for each pair of consecutive tokens, as shown in Figure 1 (the central sequence of nodes in the graph). Note that we draw the hypernym token $\boxed{\text{NN}_2}$ with a rectangle shape. We also add to the

graph a start node • and an end node ⊙, and connect them to the corresponding initial and final sentence tokens. Next, the second sentence, "In mathematics, a graph is a data structure that consists of...", is aligned to the first sentence. The alignment of the generalized sentence is perfect, apart from the $\boxed{\text{NN}_3}$ node corresponding to "data". The node is added to the graph together with the edges a$\rightarrow\boxed{\text{NN}_3}$ and $\boxed{\text{NN}_3}\rightarrow\boxed{\text{NN}_2}$. Finally, the third sentence in Table 1, "In computer science, a pixel is a dot that is part of a computer image", is generalized as "In NN NN , a ⟨TARGET⟩ is a NN". Thus, a new node NN_4 is added, corresponding to "computer" and new edges are added: In$\rightarrow\text{NN}_4$ and $\text{NN}_4\rightarrow\text{NN}_1$. Figure 1 shows the resulting WCL-1 lattice.

5 Experiments

5.1 Experimental Setup

Datasets. We conducted experiments on two different datasets:

- A corpus of 4,619 Wikipedia sentences, that contains 1,908 definitional and 2,711 non-definitional sentences. The former were obtained from a random selection of the first sentences of Wikipedia articles[3]. The defined terms belong to different Wikipedia domain categories[4], so as to capture a representative and cross-domain sample of lexical and syntactic patterns for definitions. These sentences were manually annotated with DEFINIENDUM, DEFINITOR, DEFINIENS and REST fields by an expert annotator, who also marked the hypernyms. The associated set of negative examples ("syntactically plausible" false definitions) was obtained by extracting from the same Wikipedia articles sentences in which the page title occurs.
- A subset of the ukWaC Web corpus (Ferraresi et al., 2008), a large corpus of the English language constructed by crawling the `.uk` domain of the Web. The subset includes over 300,000 sentences in which occur any of 239 terms selected from the terminology of four different domains (COMPUTER SCIENCE, ASTRONOMY, CARDIOLOGY, AVIATION).

The reason for using the ukWaC corpus is that, unlike the "clean" Wikipedia dataset, in which relatively simple patterns can achieve good results, ukWaC represents a real-world test, with many complex cases. For example, there are sentences that should be classified as definitional according to Section 3.1 but are rather uninformative, like "**dynamic programming** was the *brainchild* of an american mathematician", as well as informative sentences that are not definitional (e.g., they do not have a hypernym), like "**cubism** was characterised by muted colours and fragmented images". Even more frequently, the dataset includes sentences which are not definitions but have a definitional pattern ("A Pacific Northwest tribe's **saga** refers to a young woman who [..]"), or sentences with very complex definitional patterns ("**white body cells** are the body's clean up *squad*" and "**joule** is also an *expression of electric energy*"). These cases can be correctly handled only with fine-grained patterns. Additional details on the corpus and a more thorough linguistic analysis of complex cases can be found in Navigli et al. (2010).

Systems. For definition extraction, we experiment with the following systems:

- **WCL-1** and **WCL-3**: these two classifiers are based on our Word-Class Lattice model. WCL-1 learns from the training set a lattice for each cluster of sentences, whereas WCL-3 identifies clusters (and lattices) separately for each sentence field (DEFINIENDUM, DEFINITOR and DEFINIENS) and classifies a sentence as a definition if any combination from the three sets of lattices matches (cf. Section 3.2.4, the best combination is selected).
- **Star patterns**: a simple classifier based on the patterns learned as a result of step 1 of our WCL learning algorithm (cf. Section 3.2.1): a sentence is classified as a definition if it matches any of the star patterns in the model.
- **Bigrams**: an implementation of the bigram classifier for soft pattern matching proposed by Cui et al. (2007). The classifier selects as definitions all the sentences whose probability is above a specific threshold. The probability is calculated as a mixture of bigram and

[3] The first sentence of Wikipedia entries is, in the large majority of cases, a definition of the page title.

[4] `en.wikipedia.org/wiki/Wikipedia:Categories`

Algorithm	P	R	F_1	A
WCL-1	**99.88**	42.09	59.22	76.06
WCL-3	**98.81**	60.74	**75.23**	**83.48**
Star patterns	86.74	66.14	**75.05**	81.84
Bigrams	66.70	**82.70**	73.84	75.80
Random BL	50.00	50.00	50.00	50.00

Table 2: Performance on the Wikipedia dataset.

Algorithm	P	R†
WCL-1	**98.33**	39.39
WCL-3	**94.87**	56.57
Star patterns	44.01	**63.63**
Bigrams	46.60	45.45
Random BL	50.00	50.00

Table 3: Performance on the ukWaC dataset († Recall is estimated).

unigram probabilities, with Laplace smoothing on the latter. We use the very same settings of Cui et al. (2007), including threshold values. While the authors propose a second soft-pattern approach based on Profile HMM (cf. Section 2), their results do not show significant improvements over the bigram language model.

For hypernym extraction, we compared WCL-1 and WCL-3 with **Hearst's patterns**, a system that extracts hypernyms from sentences based on the lexico-syntactic patterns specified in Hearst's seminal work (1992). These include (hypernym in italic): "such *NP* as {NP ,} {(or | and)} NP", "NP {, NP} {,} or other *NP*", "*NP* {,} including { NP ,} {or | and} NP", "*NP* {,} especially { NP ,} {or | and} NP", and variants thereof. However, it should be noted that hypernym extraction methods in the literature do not extract hypernyms from definitional sentences, like we do, but rather from specific patterns like "X such as Y". Therefore a direct comparison with these methods is not possible. Nonetheless, we decided to implement Hearst's patterns for the sake of completeness. We could not replicate the more refined approach by Snow et al. (2004) because it requires the annotation of a possibly very large dataset of sentence fragments. In any case Snow et al. (2004) reported the following performance figures on a corpus of dimension and complexity comparable with ukWaC: the recall-precision graph indicates precision 85% at recall 10% and precision 25% at recall of 30% for the hypernym classifier. A variant of the classifier that includes evidence from coordinate terms (terms with a common ancestor in a taxonomy) obtains an increased precision of 35% at recall 30%. We see no reasons why these figures should vary dramatically on the ukWaC.

Finally, we compare all systems with the **random baseline**, that classifies a sentence as a definition with probability $\frac{1}{2}$.

Measures. To assess the performance of our systems, we calculated the following measures:

- **precision** – the number of definitional sentences correctly retrieved by the system over the number of sentences marked by the system as definitional.
- **recall** – the number of definitional sentences correctly retrieved by the system over the number of definitional sentences in the dataset.
- the **F_1-measure** – a harmonic mean of precision (P) and recall (R) given by $\frac{2PR}{P+R}$.
- **accuracy** – the number of correctly classified sentences (either as definitional or non-definitional) over the total number of sentences in the dataset.

5.2 Results and Discussion

Definition Extraction. In Table 2 we report the results of definition extraction systems on the Wikipedia dataset. Given this dataset is also used for training, experiments are performed with 10-fold cross validation. The results show very high precision for WCL-1, WCL-3 (around 99%) and star patterns (86%). As expected, bigrams and star patterns exhibit a higher recall (82% and 66%, respectively). The lower recall of WCL-1 is due to its limited ability to generalize compared to WCL-3 and the other methods. In terms of F_1-measure, star patterns and WCL-3 achieve 75%, and are thus the best systems. Similar performance is observed when we also account for negative sentences – that is we calculate accuracy (with WCL-3 performing better). All the systems perform significantly better than the random baseline.

From our Wikipedia corpus, we learned over 1,000 lattices (and star patterns). Using WCL-3, we learned 381 DF, 252 VF and 395 GF lattices, that then we used to extract definitions from

Algorithm	Full	Substring
WCL-1	**42.75**	77.00
WCL-3	40.73	**78.58**

Table 4: Precision in hypernym extraction on the Wikipedia dataset

Algorithm	Full		Substring	
WCL-1	86.19	(206)	**96.23**	(230)
WCL-3	**89.27**	(383)	**96.27**	(413)
Hearst	65.26	(62)	88.42	(84)

Table 5: Precision in hypernym extraction on the ukWaC dataset (number of hypernyms in parentheses).

the ukWaC dataset. To calculate precision on this dataset, we manually validated the definitions output by each system. However, given the large size of the test set, recall could only be estimated. To this end, we manually analyzed 50,000 sentences and identified 99 definitions, against which recall was calculated. The results are shown in Table 3. On the ukWaC dataset, WCL-3 performs best, obtaining 94.87% precision and 56.57% recall (we did not calculate F_1, as recall is estimated). Interestingly, star patterns obtain only 44% precision and around 63% recall. Bigrams achieve even lower performance, namely 46.60% precision, 45.45% recall. The reason for such bad performance on ukWaC is due to the very different nature of the two datasets: for example, in Wikipedia most "is a" sentences are definitional, whereas this property is not verified in the real world (that is, on the Web, of which ukWaC is a sample). Also, while WCL does not need any parameter tuning[5], the same does not hold for bigrams[6], whose probability threshold and mixture weights need to be best tuned on the task at hand.

Hypernym Extraction. For hypernym extraction, we tested WCL-1, WCL-3 and Hearst's patterns. Precision results are reported in Tables 4 and 5 for the two datasets, respectively. The Substring column refers to the case in which the captured hypernym is a substring of what the annotator considered to be the correct hypernym. Notice that this is a complex matter, because often the selection of a hypernym depends on semantic and contextual issues. For example, "**Fluoroscopy** is an *imaging method*" and "the **Mosaic** was an interesting *project*" have precisely the same genus pattern, but (probably depending on the vagueness of the noun in the first sentence, and of the adjective in the second) the annotator selected respectively *imaging method* and *project* as hypernyms. For the above reasons it is difficult to achieve high performance in capturing the correct hypernym (e.g. 40.73% with WCL-3 on Wikipedia). However, our performance of identifying a substring of the correct hypernym is much higher (around 78.58%). In Table 4 we do not report the precision of Hearst's patterns, as only one hypernym was found, due to the inherently low coverage of the method.

On the ukWaC dataset, the hypernyms returned by the three systems were manually validated and precision was calculated. Both WCL-1 and WCL-3 obtained a very high precision (86-89% and 96% in identifying the exact hypernym and a substring of it, respectively). Both WCL models are thus equally robust in identifying hypernyms, whereas WCL-1 suffers from a lack of generalization in definition extraction (cf. Tables 2 and 3). Also, given that the ukWaC dataset contains sentences in which any of 239 domain terms occur, WCL-3 extracts on average 1.6 and 1.7 full and substring hypernyms per term, respectively. Hearst's patterns also obtain high precision, especially when substrings are taken into account. However, the number of hypernyms returned by this method is much lower, due to the specificity of the patterns (62 vs. 383 hypernyms returned by WCL-3).

6 Conclusions

In this paper, we have presented a lattice-based approach to definition and hypernym extraction. The novelty of our approach is:

1. The use of a lattice structure to generalize over lexico-syntactic definitional patterns;
2. The ability of the system to jointly identify definitions and extract hypernyms;
3. The generality of the method, which applies to generic Web documents in any domain and style, and needs no parameter tuning;

[5]WCL has only one threshold value θ to be set for determining frequent words (cf. Section 3.1). However, no tuning was made for choosing the best value of θ.

[6]We had to re-tune the system parameters on ukWaC, since with the original settings of Cui et al. (2007) performance was much lower.

4. The high performance as compared with the best-known methods for both definition and hypernym extraction. Our approach outperforms the other systems particularly where the task is more complex, as in real-world documents (i.e., the ukWaC corpus).

Even though definitional patterns are learned from a manually annotated dataset, the dimension and heterogeneity of the training dataset ensures that training needs not to be repeated for specific domains[7], as demonstrated by the cross-domain evaluation on the ukWaC corpus.

The datasets used in our experiments are available from `http://lcl.uniroma1.it/wcl`. We also plan to release our system to the research community. In the near future, we aim to apply the output of our classifiers to the task of automated taxonomy building, and to test the WCL approach on other information extraction tasks, like hypernym extraction from generic sentence fragments, as in Snow et al. (2004).

References

Eneko Agirre, Ansa Olatz, Xabier Arregi, Xabier Artola, Arantza Daz de Ilarraza Snchez, Mikel Lersundi, David Martnez, Kepa Sarasola, and Ruben Urizar. 2000. Extraction of semantic relations from a basque monolingual dictionary using constraint grammar. In *Proceedings of Euralex*.

Claudia Borg, Mike Rosner, and Gordon Pace. 2009. Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction 2009 (wDE'09)*.

William M. Campbell, M. F. Richardson, and D. A. Reynolds. 2007. Language recognition with word lattices and support vector machines. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, pages 989–992, Honolulu, HI.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the* 37^{th} *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 120–126, Maryland, USA.

Claudio Carpineto and Giovanni Romano. 2005. Using concept lattices for text retrieval and mining. In B. Ganter, G. Stumme, and R. Wille, editors, *Formal Concept Analysis*, pages 161–179.

Christopher Collins, Bob Carpenter, and Gerald Penn. 2004. Head-driven parsing for word lattices. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 231–238, Barcelona, Spain, July.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. 1990. *Introduction to algorithms*. the MIT Electrical Engineering and Computer Science Series. MIT Press, Cambridge, MA.

Hang Cui, Min-Yen Kan, and Tat-Seng Chua. 2007. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems*, 25(2):8.

Łukasz Degórski, Michał Marcinczuk, and Adam Przepiórkowski. 2008. Definition extraction using a sequential combination of baseline grammars and machine learning classifiers. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco.

William Dolan, Lucy Vanderwende, and Stephen D. Richardson. 1993. Automatically deriving structured knowledge bases from on-line dictionaries. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, pages 5–14.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, pages 1012–1020, Columbus, Ohio, USA.

Christopher Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2009)*, pages 406–414, Boulder, Colorado, USA.

Ismail Fahmi and Gosse Bouma. 2006. Learning to identify definitions using syntactic features. In *Proceedings of the EACL 2006 workshop on Learning Structured Information in Natural Language Applications*, pages 64–71, Trento, Italy.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large Web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*, Marrakech, Morocco.

Aldo Gangemi, Roberto Navigli, and Paola Velardi. 2003. The OntoWordNet project: Extension and axiomatization of conceptual relations in WordNet. In *Proceedings of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2003)*, pages 820–838, Catania, Italy.

Rosa Del Gaudio and António Branco. 2007. Automatic extraction of definitions in portuguese: A rule-based approach. In *Proceedings of the TeMa Workshop*.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the* 14^{th} *International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France.

[7] Of course, it would need some additional work if applied to languages other than English. However, the approach does not need to be adapted to the language of interest.

Eduard Hovy, Andrew Philpot, Judith Klavans, Ulrich Germann, and Peter T. Davis. 2003. Extending metadata definitions by automatically extracting and organizing glossary definitions. In *Proceedings of the 2003 Annual National Conference on Digital Government Research*, pages 1–6. Digital Government Society of North America.

Adrian Iftene, Diana Trandabă, and Ionut Pistol. 2007. Natural language processing and knowledge representation for elearning environments. In *Proc. of Applications for Romanian. Proceedings of RANLP workshop*, pages 19–25.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008. Word lattice reranking for chineseword segmentation and part-of-speech tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 385–392, Manchester, UK.

Judith Klavans and Smaranda Muresan. 2001. Evaluation of the DEFINDER system for fully automatic glossary construction. In *Proc. of the American Medical Informatics Association (AMIA) Symposium*.

Michael Tully Klein. 2008. *Understanding English with Lattice-Learning, Master thesis*. MIT, Cambridge, MA, USA.

Lambert Mathias and William Byrne. 2006. Statistical phrase-based speech translation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, Toulouse, France.

George A. Miller, R.T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Roberto Navigli and Paola Velardi. 2006. Ontology enrichment through automatic semantic annotation of on-line glossaries. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*, pages 126–140, Podebrady, Czech Republic.

Roberto Navigli, Paola Velardi, and Juana María Ruiz-Martínez. 2010. An annotated dataset for extracting definitions and hypernyms from the Web. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Roberto Navigli. 2009a. Using cycles and quasi-cycles to disambiguate dictionary glosses. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pages 594–602, Athens, Greece.

Roberto Navigli. 2009b. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Michael P. Oakes. 2005. Using hearst's rules for the automatic acquisition of hyponyms for mining a pharmaceutical corpus. In *Proceedings of the Workshop Text Mining Research*.

Adam Przepiórkowski, Lukasz Degórski, Beata Wójtowicz, Miroslav Spousta, Vladislav Kuboň, Kiril Simov, Petya Osenova, and Lothar Lemnitzer. 2007. Towards the automatic extraction of definitions in slavic. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing (in ACL '07)*, pages 43–50, Prague, Czech Republic. Association for Computational Linguistics.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read*, pages 88–93.

Horacio Saggion. 2004. Identifying denitions in text collections for question answering. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Antonio Sanfilippo and Victor Poznański. 1992. The acquisition of lexical knowledge from combined machine-readable dictionary sources. In *Proceedings of the third Conference on Applied Natural Language Processing*, pages 80–87.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Josh Schroeder, Trevor Cohn, and Philipp Koehn. 2009. Word lattices for multi-source translation. In *Proceedings of the European Chapter of the Association for Computation Linguistics (EACL 2009)*, pages 719–727, Athens, Greece.

Rion Snow, Dan Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1297–1304.

Angelika Storrer and Sandra Wellinghoff. 2006. Automated detection and annotation of term definitions in german text corpora. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, Italy.

Paola Velardi, Roberto Navigli, and Pierluigi D'Amadio. 2008. Mining the Web to create specialized glossaries. *IEEE Intelligent Systems*, 23(5):18–25.

Eline Westerhout and Paola Monachesi. 2007. Extraction of dutch definitory contexts for eLearning purposes. In *Proceedings of CLIN*.

Eline Westerhout. 2009. Definition extraction using linguistic and structural features. In *Proceedings of the RANLP 2009 Workshop on Definition Extraction*, pages 61–67.

Chunxia Zhang and Peng Jiang. 2009. Automatic extraction of definitions. In *Proceedings of 2nd IEEE International Conference on Computer Science and Information Technology*, pages 364–368.

Zhao-man Zhong, Zong-tian Liu, and Yan Guan. 2008. Precise information extraction from text based on two-level concept lattice. In *Proceedings of the 2008 International Symposiums on Information Processing (ISIP '08)*, pages 275–279, Washington, DC, USA.

On Learning Subtypes of the Part-Whole Relation: Do Not Mix your Seeds

Ashwin Ittoo
University of Groningen
Groningen, The Netherlands
r.a.ittoo@rug.nl

Gosse Bouma
University of Groningen
Groningen, The Netherlands
g.bouma@rug.nl

Abstract

An important relation in information extraction is the part-whole relation. Ontological studies mention several types of this relation. In this paper, we show that the traditional practice of initializing minimally-supervised algorithms with a single set that mixes seeds of different types fails to capture the wide variety of part-whole patterns and tuples. The results obtained with mixed seeds ultimately converge to one of the part-whole relation types. We also demonstrate that all the different types of part-whole relations can still be discovered, regardless of the type characterized by the initializing seeds. We performed our experiments with a state-of-the-art information extraction algorithm.

1 Introduction

A fundamental semantic relation in many disciplines such as linguistics, cognitive science, and conceptual modelling is the part-whole relation, which exists between parts and the wholes they compise (Winston et al., 1987; Gerstl and Pribbenow, 1995). Different types of part-whole relations, classified in various taxonomies, are mentioned in literature (Winston et al., 1987; Odell, 1994; Gerstl and Pribbenow, 1995; Keet and Artale, 2008). The taxonomy of Keet and Artale (2008), for instance, distinguishes part-whole relations based on their transitivity, and on the semantic classes of entities they sub-categorize. Part-whole relations are also crucial for many information extraction (IE) tasks (Girju et al., 2006). Annotated corpora and semantic dictionaries used in IE, such as the ACE corpus[1] and WordNet (Fellbaum, 1998), include examples of part-whole relations. Also, previous relation extraction work, such as Berland and Charniak (1999) and Girju et al. (2006), have specifically targeted the discovery of part-whole relations from text. Furthermore, part-whole relations are de-facto benchmarks for evaluating the performance of general relation extraction systems (Pantel and Pennacchiotti, 2006; Beamer et al., 2008; Pyysalo et al., 2009). However, these relation extraction efforts have overlooked the ontological distinctions between the different types of part-whole relations. They assume the existence of a single relation, subsuming the different part-whole relation types.

In this paper, we show that enforcing the ontological distinctions between the different types of part-whole relations enable information extraction systems to capture a wider variety of both generic and specialised part-whole lexico-syntactic patterns and tuples. Specifically, we address 3 major questions.

1. Is information extraction (IE) harder when learning the individual types of part-whole relations? That is, we determine whether the performance of state-of-the-art IE systems in learning the individual part-whole relation types increases (due to more coherency in the relations' linguistic realizations) or drops (due to fewer examples), compared to the traditional practice of considering a single part-whole relation.

2. Are the patterns and tuples discovered when focusing on a specific part-whole relation type confined to that particular type? That is, we investgate whether IE systems discover examples representative of the different types by targetting one particular part-whole relation type.

3. Are more distinct examples discovered when IE systems learn the individual part-whole relation types? That is, we determine whether

[1] http://projects.ldc.upenn.edu/ace/

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1328–1336,
Uppsala, Sweden, 11-16 July 2010.

a wider variety of unique patterns and tuples are extracted when IE systems target the different types of part-whole relations instead of considering a single part-whole relation that subsumes all the different types.

To answer these questions, we bootstrapped a minimally-supervised relation extraction algorithm, based on Espresso (Pantel and Pennacchiotti, 2006), with different seed-sets for the various types of part-whole relations, and analyzed the harvested tuples and patterns.

2 Previous Work

Investigations on the part-whole relations span across many disciplines, such as conceptual modeling (Artale et al., 1996; Keet, 2006; Keet and Artale, 2008), which focus on the ontological aspects, and linguistics and cognitive sciences, which focus on natural language semantics. Several linguistically-motivated taxonomies (Odell, 1994; Gerstl and Pribbenow, 1995), based on the work of Winston et al. (1987), have been proposed to clarify the semantics of the different part-whole relations types across these various disciplines. Keet and Artale (2008) developed a formal taxonomy, distinguishing transitive *mereological* part-whole relations from intransitive *meronymic* ones. Meronymic relations identified are: 1) *member-of*, between a physical object (or role) and an aggregation, e.g. *player-team*, 2) *constituted-of*, between a physical object and an amount of matter e.g. *clay-statue*, 3) *sub-quantity-of*, between amounts of matter or units, e.g. *oxygen-water* or *m-km*, and 4)*participates-in*, between an entity and a process e.g. *enzyme-reaction*. Mereological relations are: 1)*involved-in*, between a phase and a process, e.g. *chewing-eating*, 2) *located-in*, between an entity and its 2-dimensional region, e.g. *city-region*, 3)*contained-in*, between an entity and its 3-dimensional region, e.g.*tool-trunk*, and 4)*structural part-of*, between integrals and their (functional) components, e.g. *engine-car*. This taxonomy further discriminates between part-whole relation types by enforcing semantical selectional restrictions, in the form of DOLCE ontology (Gangemi et al., 2002) classes, on their entities.

In NLP, information extraction (IE) techniques, for discovering part-whole relations from text have also been developed. Berland and Charniak (1999) use manually-crafted patterns, similar to Hearst (1992), and on initial "seeds" denoting "whole" objects (e.g. building) to harvest possible "part" objects (e.g. room) from the North Americal News Corpus (NANC) of 1 million words. They rank their results with measures like log-likelihood (Dunning, 1993), and report a maximum accuracy of 70% over their top-20 results. In the supervised approaches in Girju et al. (2003) and Girju et al. (2006), lexical patterns expressing part-whole relations between WordNet concept pairs are manually extracted from 20,000 sentences of the L.A Times and SemCor corpora (Miller et al., 1993), and used to generate a training corpus, with manually-annotated positive and negative examples of part-whole relations. Classification rules, induced over the training data, achieve a precision of 80.95% and recall of 75.91% in predicting whether an unseen pattern encode a part-whole relation. Van Hage et al. (2006) acquire 503 part-whole pairs from dedicated thesauri (e.g. AGROVOC[2]) to learn 91 reliable part-whole patterns. They substituted the patterns' "part" arguments with known entities to formulate web-search queries. Corresponding "whole" entities were then discovered from documents in the query results with a precision of 74%. The part-whole relation is also a benchmark to evaluate the performance of general information extraction systems. The Espresso algorithm (Pantel and Pennacchiotti, 2006) achieves a precision of 80% in learning part-whole relations from the Acquaint (TREC-9) corpus of nearly 6M words. Despite the reasonable performance of the above IE systems in discovering part-whole relations, they overlook the ontological distinctions between the different relation types. For example, Girju et al. (2003) and Girju et al. (2006) assume a single part-whole relation, encompassing all the different types mentioned in the taxonomy of Winston et al. (1987). Similarly, the minimally-supervised Espresso algorithm (Pantel and Pennacchiotti, 2006) is initialized with a single set that mixes seeds of heterogeneous types, such as *leader-panel* and *oxygen-water*, which respectively correspond to the *member-of* and *sub-quantity-of* relations in the taxonomy of Keet and Artale (2008).

[2] http://aims.fao.org/website/AGROVOC-Thesaurus/sub

3 Methodology

Our aim is to compare the relations harvested when a minimally-supervised IE algorithm is initialized with separate sets of seeds for each type of part-whole relation, and when it is initialized following the traditional practice of a single set that mixes seeds of the different types. To distinguish between types of part-whole relations, we commit to the taxonomy of Keet and Artale (2008) (Keet's taxonomy), which uses sound ontological formalisms to unambiguously discrimate the relation types. Also, this taxonomy classifies the various part-whole relations introduced in literature, including ontologically-motivated mereological relations and linguistically-motivated meronymic ones. We adopt a 3-step approach to address our questions from section 1.

1. Define prototypical seeds (part-whole tuples) as follows:
   - (Separate) sets of seeds for each type of part-whole relation in Keet's taxonomy.
   - A single set that mixes seeds denoting all the different part-whole relations types.
2. Part-whole relations extraction from a corpus by initializing a minimally-supervised IE algorithm with the seed-sets
3. Evaluation of the harvested relations to determine performance gain/loss, types of part-whole relations extracted, and distinct and unique patterns and tuples discovered.

The corpora and IE algorithm we used, and the seed-sets construction are described below. Results are presented in the next section.

3.1 Corpora

We used the English and Dutch Wikipedia texts since their broad-coverage and size ensures that they include sufficient lexical realizations of the different types of part-whole relations. Wikipedia has also been targeted by recent IE efforts (Nguyen et al., 2007; Wu and Weld, 2007). However, while they exploited the structured features (e.g. infoboxes), we only consider the unstructured texts. The English corpus size is approximately 470M words ($\sim$ 80% of the August 2007 dump), while for Dutch, we use the full text collection (February 2008 dump) of approximately 110M words. We parsed the English and Dutch corpora respectively with the Stanford[3] (Klein and Manning, 2003) and the Alpino[4] (van Noord, 2006) parsers, and formalized the relations between terms (entities) as dependency paths. A dependency path is the shortest path of lexico-syntactic elements, i.e. shortest lexico-syntactic pattern, connecting entities (proper and common nouns) in their parse-trees. Such a formalization has been successfully employed in previous IE tasks (see Stevenson and Greenwood (2009) for an overview). Compared to traditional surface-pattern representations, used by Pantel and Pennacchiotti (2006), dependency paths abstract from surface texts to capture long range dependencies between terms. They also alleviate the manual authoring of large numbers of surface patterns. In our formalization, we substitute entities in the dependency paths with generic placeholders PART and WHOLE. Below, we show two dependency paths (1-b) and (2-b), respectively derived from English and Dutch Wikipedia sentences (1-a) and (2-a), and denoting the relations between *sample-song*, and *alkaloïde-plant*.

(1) a. The song "Mao Tse Tung Said" by Alabama 3 contains samples of a speech by Jim Jones
    b. WHOLE+nsubj ← contains → dobj+PART

(2) a. Alle delen van de planten bevatten alkaloïden en zijn daarmee giftig (*All parts of the plants contain alkaloids and therefore are poisonous*)
    b. WHOLE+obj1+van+mod+deel+su ← bevat→ obj1+PART

In our experiments, we only consider those entity pairs (tuples), patterns, and co-occuring pairs-patterns with a minimum frequency of 10 in the English corpus, and 5 in the Dutch corpus. Statistics on the number of tuples and patterns preserved after applying the frequency cut-off are given in Table 1.

3.2 Information Extraction Algorithm

As IE algorithm for extracting part-whole relations from our texts, we relied on *Espresso*, a minimally-supervised algorithm, as described by Pantel and Pennacchiotti (2006). They show

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://www.let.rug.nl/~vannoord/alp/Alpino

	English	Dutch
words	470.0	110.0
pairs	328.0	28.8
unique pairs	6.7	1.4
patterns	238.0	54.0
unique patterns	2.0	0.9

Table 1: Corpus Statistics in millions

that the algorithm achieves state-of-the-art performance when initialized with relatively small seed-sets over the Acquaint corpus ($\sim$ 6M words). Recall is improved with web search queries as additional source of information.

Espresso extracts surface patterns connecting the seeds (tuples) in a corpus. The reliability of a pattern p, $r(p)$, given a set of input tuples I, is computed using (3), as its average strength of association with each tuple,i, weighted by each tuple's reliability, $r_\iota(i)$.

$$r_\pi(p) = \frac{\sum_{i \in I} \left(\frac{pmi(i,p)}{\max_{pmi}} \times r_\iota(i) \right)}{|I|} \quad (3)$$

In this equation, $pmi(i,p)$ is the pointwise mutual information score (Church and Hanks, 1990) between a pattern, p (e.g. *consist-of*), and a tuple, i (e.g. *engine-car*), and max_{pmi} is the maximum PMI score between all patterns and tuples. The reliability of the initializing seeds is set to 1.

The top-k most reliable patterns are selected to find new tuples. The reliability of each tuple i, $r_\iota(i)$ is computed according to (4), where P is the set of harvested patterns. The top-m most reliable tuples are used to infer new patterns.

$$r_\iota(i) = \frac{\sum_{i \in I} \left(\frac{pmi(i,p)}{\max_{pmi}} \times r_\pi(p) \right)}{|P|} \quad (4)$$

The recursive discovery of patterns from tuples and vice-versa is repeated until a threshold number of patterns and/or tuples have been extracted. In our implementation, we maintain the core of the original Espresso algorithm, which pertains to estimating the reliability of patterns and tuples.

Pantel and Pennacchiotti (2006) mention that their method is independent of the way patterns are formulated. Thus, instead of relying on surface patterns, we use dependency paths (as described above). Another difference is that while Pantel and Pennacchiotti (2006) complement their small corpus with documents retrieved from the web, we only rely on patterns extracted from our (much larger) corpora. Finally, we did not apply the discounting factor suggested in Pantel and Pennacchiotti (2006) to correct for the fact that PMI overestimates the importance of low-frequency events. Instead, as explained above, we applied a general frequency cut-off.[5]

3.3 Seed Selection

Initially,we selected seeds from WordNet (Fellbaum, 1998) (for English) and EuroWordNet (Vossen, 1998) (for Dutch) to initialize the IE algorithm. However, we found that these pairs, such as *acinos-mother of thyme* or *radarscherm-radarapparatuur (radar screen - radar equipment*, hardly co-occured with reasonable frequency in Wikipedia sentences, hindering pattern extraction. We therefore adopted the following strategy.

We searched our corpora for archetypal patterns, e.g. *contain* , which characterize all the different types of part-whole relations. The tuples sub-categorized by these patterns in the English texts were automatically[6] typed to appropriate DOLCE ontology[7] classes, corresponding to those employed by Keet and Artale for constraining the entity pairs participating in different types of part-whole relations. The types of part-whole relations instantiated by the tuples could then be determined based on their ontological classes. Separate sets of 20 tuples, with each set corresponding to a specific relation type in the taxonomy of Keet and Artale (Keet's taxonomy), were then created. For example, the English Wikipedia tuple $t1$ =*actor-cast* was used as a seed to discover *member-of* part-whole relations since both its elements were typed to the SOCIAL OBJECT class of the DOLCE ontology, and according to Keet's taxonomy, they instantiate a *member-of* relation. Seeds for extracting relations from the Dutch corpus were defined in a similar way, except that we manually determined their ontological classes based on the class glossary of DOLCE.

Below, we only report on the *member-of* and *sub-quantity-of* meronymic relations, and on the *located-in*, *contained-in* and *structural part-of* mereological relations. We were unable to find sufficient seeds for the *constituted-of* meronymic

[5] We experimented with the suggested discounting factor for PMI, but were not able to improve over the accuracy scores reported later.

[6] Using the Java-OWL API, from `http://protege.stanford.edu/plugins/owl/api/`

[7] OWL Version 0.72, downloaded from `http://www.loa-cnr.it/DOLCE.html/`

Lg	Part	Whole	#	Type
EN	grave	church	155	contain
NL	beeld	kerk	120	contain
	(*statue*)	(*church*)		
EN	city	region	3735	located
NL	abdij	gemeente	36	located
	(*abbey*)	(*community*)		
EN	actor	cast	432	member
NL	club	voetbal_bond	178	member
	(*club*)	(*soccer union*)		
EN	engine	car	3509	structural
NL	geheugen	computer	14	structural
	(*memory*)	(*computer*)		
EN	alcohol	wine	260	subquant
NL	alcohol	bier	28	subquant
	(*alcohol*)	(*beer*)		

Table 2: Seeds used for learning part-whole relations (*contained-in, located-in, member-of, structural part-of, sub-quantity-of*).

relations (e.g. *clay-statue*). Also, we did not experiment with the *participates-in* and *involved-in* relations since their lexical realizations in our corpora are sparse, and they contain at least one verbal argument, whereas we only targeted patterns connecting nomimals. Sample seeds, their corpus frequency, and the part-whole relation type they instantiate from the English (EN) and Dutch (NL) corpora are illustrated in Table 2. Besides the five specialized seed-sets of 20 prototypical tuples for the aforementioned relations, we also defined a *general* set of mixed seeds, which combines four seeds from each of the specialized sets.

4 Experiments and Evaluation

We initialized our IE algorithm with the seed-sets to extract part-whole relations from our corpora. The same parameters as Pantel and Pennacchiotti (2006) were used. That is, the 10 most reliable patterns inferred from the initial seeds are bootstrapped to induce 100 part-whole tuples. In each subsequent iteration, we learn one additional pattern and 100 additional tuples. We evaluated our results after 5 iterations since the performance in later iterations was almost constant. The results are discussed next.

	meronomic		mereological			
	memb	subq	cont	struc	locat	gen
EN	0.67	0.74	0.70	0.82	0.75	0.80
NL	0.68	0.60	0.60	0.60	0.70	0.71

Table 3: Precision for seed-sets representing specific types of part-whole relations (*member-of, sub-quantity-of, contained-in, structural part-of* and *located-in*), and for the *general* set composed of all types.

4.1 Precision of Extracted Relations

Two human judges manually evaluated the tuples extracted from the English and Dutch corpora per seed-set in each iteration of our algorithm. Tuples that unambiguously instantiated part-whole relations were considered *true positives*. Those that did not were considered *false positives*. Ambiguous tuples were discarded. The precision of the tuples discovered by the different seed-sets in the last iteration of our algorithm are in Table 3.

These results reveal that the precision of harvested tuples varies depending on the part-whole relation type that the initializing seeds denote. Mereological seeds (*cont, struct, locat* sets) outperformed their meronymic counterparts (*memb, subq*) in extracting relations with higher precision from the English texts. This could be attributed to their formal ontological grounding, making them less ambiguous than the linguistically-motivated meronymic relations (Keet, 2006; Keet and Artale, 2008). The precision variations were less discernible for tuples extracted from the Dutch corpus, although the best precision was still achieved with mereological *located-in* seeds. We also noticed that the precision of tuples extracted from both the English and Dutch corpora by the general set of mixed seeds was as high as the maximum precision obtained by the individual sets of specialized seeds over these two corpora, i.e. 0.80 (*general* seeds) vs. 0.82 (*structural part-of* seeds) for English, and 0.71 (*general* seeds) vs. 0.70 (*located-in* seeds) for Dutch. Based on these findings, we address our first question, and conclude that 1) the type of relation instantiated by the initializing seeds affects the performance of IE algorithms, with mereological seeds being in general more fertile than their meronymic counterparts, and generating higher-precision tuples; 2) the precision achieved when initializing IE algorithms with a general set, which mixes

seeds of heterogeneous part-whole relation types, is comparable to the best results obtained with individual sets of specialized seeds, denoting specific part-whole relations. An evaluation of the patterns and tuples extracted indicated considerable precision drop between successive iterations of our algorithm. This appears to be due to *semantic drift* (McIntosh and Curran, 2009), where highly-ambiguous patterns promote incorrect tuples , which in turn, compound the precision loss.

4.2 Types of Extracted Relations

Initializing our algorithm with seeds of a particular type always led to the discovery of tuples characterizing other types of part-whole relations in the English corpus. This can be explained by prototypical patterns, e.g. "include", generated regardless of the seeds' types, and which are highy correlated with, and hence, trigger tuples denoting other part-whole relation types. An almost similar observation was made for the Dutch corpus, except that tuples instantiating the *member-of* relation could only be learnt using initial seeds of that particular type (i.e. *member-of*). Upon inspecting our results, it was found that this phenomenon was due to the distinct and specific patterns, such as "treedt toe tot" ("become member of"), which linguistically realize the *member-of* relations in the Dutch corpus. Thus, initializing our IE algorithm with seeds that instantiate relations other than *member-of* fails to detect these unique patterns, and fails to subsequently discover part-whole tuples describing the *member-of* relations. Our findings are illustrated in Table 4, where each cell lists a tuple of a particular type (column), which was harvested from seeds of a given type (row). These results answer our second question.

4.3 Distinct Patterns and Tuples

We address our third question by comparing the output of our algorithm to determine whether the results obtained by initializing with the individual specialized seeds were (dis)similar and/or distinct. Each result set consisted of maximally 520 tuples (including 20 initializing seeds) and 15 lexico-syntactic patterns, obtained after five iterations.

Tuples extracted from the English corpus using the *member-of* and *contained-in* seed-sets exhibited a high degree of similarity, with 465 common tuples discovered by both sets. These identical tuples were also assigned the same ranks (reliability) in the results generated by the *member-of* and *contained-in* seeds, with a Spearman rank correlation of 0.82 between their respective outputs. This convergence was also reflected in the fact that the *member-of* and *contained-in* seeds generated around 80% of common patterns. These patterns were mostly prototypical ones indicative of part-whole relations, such as WHOLE+nsubj ← include → dobj+PART ("include") and their cognates involving passive forms and relative clauses. However, the specialized seeds also generated distinct patterns, like "joined as" and "released with" for the *member-of* and *contained-in* seeds respectively.

The most distinct tuples and patterns were harvested with the *sub-quantity-of*, *structural part-of*, and *located-in* seeds. Negative Spearman correlation scores were obtained when comparing the results of these three sets among themselves, and with the results of the *member-of* and *contained-in* seeds, indicating insignificant similarity and overlap. Examining the patterns harvested by the *sub-quantity-of*, *structural part-of*, and *located-in* seeds revealed a high prominence of specialized and unique patterns, which specifically characterize these relations. Examples of such patterns include "made with", "released with" and "found in", which lexically realize the *sub-quantity-of*, *structural part-of*, and *located-in* relations respectively.

For the Dutch corpus, the seeds that generated the most similar tuples were those corresponding to the *sub-quantity-of*, *contained-in*, and *structural part-of* relations, with 490 common tuples discovered, and a Spearman rank correlation in the range of 0.89-0.93 between their respective outputs. As expected, these seeds also led to the discovery of a substantial number of common and prototypical part-whole patterns. Examples include "bevat" ("contain"), "omvat" ("comprise"), and their variants. The most distinct results were harvested by the *located-in* and *member-of* seeds, with negative Spearman correlation scores between the output tuples indicating hardly any overlap. We also found out that the patterns harvested by the *located-in* and *member-of* seeds characteristically pertained to these relations. Example of such patterns include "ligt in" ("lie in"), "is gelegen in" ("is located in"), and "treedt toe tot" ("become member of"), respectively describing the *located-in* and *member-of* relations.

Thus, we observed that 1) tuples harvested from

	Tuples→ *Seeds*↓	meronomic member	meronomic subquant	mereological contained	mereological struct	mereological located
EN	member	ship-convoy	alcohol-wine	card-deck	proton-nucleus	lake-park
	subquant	aircraft-fleet	moisture-soil	building-complex	engine-car	commune-canton
	contained	aircraft-fleet	alcohol-wine	relic-church	base-spacecraft	campus-city
	structural	brother-family	mineral-bone	library-building	inlay-fingerboard	hamlet-town
	located	performer-cast	alcohol-blood	artifact-museum	chassis-car	city-shore
NL	member	sporter-ploeg (*athlete-team*)	helium-atmosfeer (*helium-atmosphere*)	stalagmieten-grot (*stalagnites-cave*)	shirt-tenue (*shirt-outfit*)	boerderij-dorp (*farm-village*)
	subquant	—	vet-kaas (*fat-cheese*)	pijp_orgel-kerk (*pipe-organ-church*)	kam-gitaar (*bridge-guitar*)	paleis-stad (*palace-city*)
	contained	—	tannine-wijn (*tannine-wine*)	kamer-toren (*room-tower*)	atoom-molecule (*atom-molecule*)	paleis-stad (*palace-city*)
	structural	—	kinine-tonic (*quinine-tonic*)	beeld-kerk *statue-church*)	wervel-ruggengraat (*vertebra-backbone*)	paleis-stad (*palace-city*)
	located	—	—	kunst_werk-kathedraal (*work of art-cathedral*)	poort-muur (*gate-wall*)	metro_station-wijk (*metro station-quarter*)

Table 4: Sample tuples found per relation type.

both the English and Dutch corpora by seeds instantiating a single particular type of part-whole relation highly correlated with tuples discovered by at least one other type of seeds (*member-of* and *contained-in* for English, and *sub-quantity-of*, *contained-in* and *structural part-of* for Dutch); 2) some part-whole relations are manifested by a wide variety of specialized patterns (*sub-quantity-of*, *structural part-of*, and *located-in* for English, and *located-in* and *member-of* for Dutch).

Finally, instead of a single set that mixes seeds of different types, we created five such *general* sets by picking four different seeds from each of the specialized sets, and used them to initialize our algorithm. When examining the results of each of the five *general* sets, we found out that they were unstable, and always correlated with the output of a different specialized set.

Based on these findings, we believe that the traditional practice of initializing IE algorithms with *general* sets that mix seeds denoting different part-whole relation types leads to inherently unstable results. As we have shown, the relations extracted by combining seeds of heterogeneous types almost always converge to one specific part-whole relation type, which cannot be conclusively predicted. Furthermore, *general* seeds are unable to capture the specific and distinct patterns that lexically realize the individual types of part-whole relations.

5 Conclusions

In this paper, we have investigated the effect of ontologically-motivated distinctions in part-whole relations on IE systems that learn instances of these relations from text.

We have shown that learning from specialized seeds-sets, denoting specific types of the part-whole relations, results in precision that is as high as or higher than the precision achieved with a general set that mixes seeds of different types. By comparing the outputs generated by different seed-sets, we observed that the tuples learnt with seeds denoting a specific part-whole relation type are not confined to that particular type. In most case, we are still able to discover tuples across all the different types of part-whole relations, regardless of the type instantiated by the initializing seeds. Most importantly, we demonstrated that IE algorithms initialized with general sets of mixed seeds harvest results that tend to converge towards a specific type of part-whole relation. Conversely, when starting with seeds representing a specific type, it is likely to discover tuples and patterns that are completely distinct from those found by a mixed seed-set.

Our results also illustrate that the outputs of IE algorithms are heavily influenced by the initializing seeds, concurring with the findings of McIntosh and Curran (2009). We believe that our results show a drastic form of this phenomenon: given a set of mixed seeds, denoting heterogeneous relations, the harvested tuples may converge towards any of the relations instantiated by the seeds. Predicting the convergent relation is in usual cases impossible, and may depend on factors pertaining to corpus characteristics. This instability strongly suggests that seeds instantiating different types of relations should not be mixed, partic-

ularly when learning part-whole relations, which are characterized by many subtypes. Seeds should be defined such that they represent an ontologically well-defined class, for which one may hope to find a coherent set of extraction patterns.

Acknowledgement

Ashwin Ittoo is part of the project "Merging of Incoherent Field Feedback Data into Prioritized Design Information (DataFusion)" (`http://www.iopdatafusion.org//`), sponsored by the Dutch Ministry of Economic Affairs under the IOP-IPCR program.

Gosse Bouma acknowledges support from the Stevin LASSY project (`www.let.rug.nl/~vannoord/Lassy/`).

References

A. Artale, E. Franconi, N. Guarino, and L. Pazzi. 1996. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, 20(3):347–383.

B. Beamer, A. Rozovskaya, and R. Girju. 2008. Automatic semantic relation extraction with multiple boundary generation. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 2*, pages 824–829. AAAI Press.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

K.W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):74.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT, Cambridge.

A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. 2002. Sweetening ontologies with DOLCE. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, Lecture Notes in Computer Science*, pages 223–233.

P. Gerstl and S. Pribbenow. 1995. Midwinters, end games, and body parts: a classification of part-whole relations. *International Journal of Human Computer Studies*, 43:865–890.

R. Girju, A. Badulescu, and D. Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of HLT/NAACL*, volume 3, pages 80–87.

R. Girju, A. Badulescu, and D. Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics Morristown, NJ, USA.

C.M. Keet and A. Artale. 2008. Representing and reasoning over a taxonomy of part–whole relations. *Applied Ontology*, 3(1):91–110.

C.M. Keet. 2006. Part-whole relations in object-role models. *On the Move to Meaningful Internet Systems 2006, Lecture Notes in Computer Science*, 4278:1118–1127.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics Morristown, NJ, USA.

T. McIntosh and J.R. Curran. 2009. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Confe rence on Natural Language Processing of the AFNLP*, pages 396–404.

G.A. Miller, C. Leacock, R. Tengi, and R.T. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA workshop on Human Language Technology*, pages 303–308. New Jersey.

D.P.T. Nguyen, Y. Matsuo, and M. Ishizuka. 2007. Relation extraction from wikipedia using subtree mining. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1414. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

J. Odell. 1994. Six different kinds of composition. *Journal of Object-Oriented Programming*, 5(8):10–15.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06)*, pages 113–120, Sydney, Australia.

S. Pyysalo, T. Ohta, J.D. Kim, and J. Tsujii. 2009. Static relations: a piece in the biomedical information extraction puzzle. In *Proceedings of the Workshop on BioNLP*, pages 1–9. Association for Computational Linguistics.

Mark Stevenson and Mark Greenwood. 2009. Dependency pattern models for information extraction. *Research on Language and Computation*, 3:13–39.

W.R. Van Hage, H. Kolb, and G. Schreiber. 2006. A method for learning part-whole relations. *The Semantic Web - ISWC 2006, Lecture Notes in Computer Science*, 4273:723–735.

Gertjan van Noord. 2006. At last parsing is now operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pages 20–42. Presses univ. de Louvain.

P. Vossen, editor. 1998. *EuroWordNet A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic publishers.

M.E. Winston, R. Chaffin, and D. Herrmann. 1987. A taxonomy of part-whole relations. *Cognitive science*, 11(4):417–444.

F. Wu and D.S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Understanding the Semantic Structure of Noun Phrase Queries

Xiao Li
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
xiaol@microsoft.com

Abstract

Determining the semantic intent of web queries not only involves identifying their semantic class, which is a primary focus of previous works, but also understanding their semantic structure. In this work, we formally define the semantic structure of noun phrase queries as comprised of *intent heads* and *intent modifiers*. We present methods that automatically identify these constituents as well as their semantic roles based on Markov and semi-Markov conditional random fields. We show that the use of semantic features and syntactic features significantly contribute to improving the understanding performance.

1 Introduction

Web queries can be considered as implicit questions or commands, in that they are performed either to find information on the web or to initiate interaction with web services. Web users, however, rarely express their intent in full language. For example, to find out "what are the movies of 2010 in which johnny depp stars", a user may simply query "johnny depp movies 2010". Today's search engines, generally speaking, are based on matching such keywords against web documents and ranking relevant results using sophisticated features and algorithms.

As search engine technologies evolve, it is increasingly believed that search will be shifting away from "ten blue links" toward understanding intent and serving objects. This trend has been largely driven by an increasing amount of structured and semi-structured data made available to search engines, such as relational databases and semantically annotated web documents. Searching over such data sources, in many cases, can offer more relevant and essential results compared with merely returning web pages that contain query keywords. Table 1 shows a simplified view of a structured data source, where each row represents a movie object. Consider the query "johnny depp movies 2010". It is possible to retrieve a set of movie objects from Table 1 that satisfy the constraints *Year = 2010* and *Cast* $\ni$ *Johnny Depp*. This would deliver direct answers to the query rather than having the user sort through list of keyword results.

In no small part, the success of such an approach relies on robust understanding of query intent. Most previous works in this area focus on query intent classification (Shen et al., 2006; Li et al., 2008b; Arguello et al., 2009). Indeed, the intent class information is crucial in determining if a query can be answered by any structured data sources and, if so, by which one. In this work, we go one step further and study the semantic structure of a query, *i.e.*, individual constituents of a query and their semantic roles. In particular, we focus on *noun phrase* queries. A key contribution of this work is that we formally define query semantic structure as comprised of *intent heads* (IH) and *intent modifiers* (IM), *e.g.*,

$$[_{\text{IM:}\textit{Title}}\ \text{alice in wonderland}]\ [_{\text{IM:}\textit{Year}}\ 2010]\ [_{\text{IH}}\ \text{cast}]$$

It is determined that "cast" is an IH of the above query, representing the essential information the user intends to obtain. Furthermore, there are two IMs, "alice in wonderland" and "2010", serving as filters of the information the user receives.

Identifying the semantic structure of queries can be beneficial to information retrieval. Knowing the semantic role of each query constituent, we

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1337–1345,
Uppsala, Sweden, 11-16 July 2010.

Title	Year	Genre	Director	Cast	Review
Precious	2009	Drama	Lee Daniels	Gabby Sidibe, Mo'Nique,...	
2012	2009	Action, Sci Fi	Roland Emmerich	John Cusack, Chiwetel Ejiofor,...	
Avatar	2009	Action, Sci Fi	James Cameron	Sam Worthington, Zoe Saldana,...	
The Rum Diary	2010	Adventure, Drama	Bruce Robinson	Johnny Depp,Giovanni Ribisi,...	
Alice in Wonderland	2010	Adventure, Family	Tim Burton	Mia Wasikowska, Johnny Depp,...	

Table 1: A simplified view of a structured data source for the *Movie* domain.

can reformulate the query into a structured form or reweight different query constituents for structured data retrieval (Robertson et al., 2004; Kim et al., 2009; Paparizos et al., 2009). Alternatively, the knowledge of IHs, IMs and semantic labels of IMs may be used as additional evidence in a learning to rank framework (Burges et al., 2005).

A second contribution of this work is to present methods that automatically extract the semantic structure of noun phrase queries, *i.e.*, IHs, IMs and the semantic labels of IMs. In particular, we investigate the use of transition, lexical, semantic and syntactic features. The semantic features can be constructed from structured data sources or by mining query logs, while the syntactic features can be obtained by readily-available syntactic analysis tools. We compare the roles of these features in two discriminative models, Markov and semi-Markov conditional random fields. The second model is especially interesting to us since in our task it is beneficial to use features that measure segment-level characteristics. Finally, we evaluate our proposed models and features on manually-annotated query sets from three domains, while our techniques are general enough to be applied to many other domains.

2 Related Works

2.1 Query intent understanding

As mentioned in the introduction, previous works on query intent understanding have largely focused on classification, *i.e.*, automatically mapping queries into semantic classes (Shen et al., 2006; Li et al., 2008b; Arguello et al., 2009). There are relatively few published works on understanding the semantic structure of web queries. The most relevant ones are on the problem of query tagging, *i.e.*, assigning semantic labels to query terms (Li et al., 2009; Manshadi and Li, 2009). For example, in "canon powershot sd850 camera silver", the word "canon" should be tagged as *Brand*. In particular, Li et al. leveraged click-through data and a database to automatically derive training data for learning a CRF-based tagger. Manshadi and Li developed a hybrid, generative grammar model for a similar task. Both works are closely related to one aspect of our work, which is to assign semantic labels to IMs. A key difference is that they do not conceptually distinguish between IHs and IMs.

On the other hand, there have been a series of research studies related to IH identification (Pasca and Durme, 2007; Pasca and Durme, 2008). Their methods aim at extracting attribute names, such as *cost* and *side effect* for the concept *Drug*, from documents and query logs in a weakly-supervised learning framework. When used in the context of web queries, attribute names usually serve as IHs. In fact, one immediate application of their research is to understand web queries that request factual information of some concepts, *e.g.* "asiprin cost" and "aspirin side effect". Their framework, however, does not consider the identification and categorization of IMs (attribute values).

2.2 Question answering

Query intent understanding is analogous to question understanding for *question answering* (QA) systems. Many web queries can be viewed as the keyword-based counterparts of natural language questions. For example, the query "california national" and "national parks califorina" both imply the question "What are the national parks in California?". In particular, a number of works investigated the importance of *head noun* extraction in understanding *what-type* questions (Metzler and Croft, 2005; Li et al., 2008a). To extract head nouns, they applied syntax-based rules using the information obtained from part-of-speech (POS) tagging and deep parsing. As questions posed in natural language tend to have strong syntactic structures, such an approach was demonstrated to be accurate in identifying head nouns.

In identifying IHs in noun phrase queries, however, direct syntactic analysis is unlikely to be as effective. This is because syntactic structures are in general less pronounced in web queries. In this

work, we propose to use POS tagging and parsing outputs as features, in addition to other features, in extracting the semantic structure of web queries.

2.3 Information extraction

Finally, there exist large bodies of work on information extraction using models based on Markov and semi-Markov CRFs (Lafferty et al., 2001; Sarawagi and Cohen, 2004), and in particular for the task of named entity recognition (McCallum and Li, 2003).

The problem studied in this work is concerned with identifying more generic "semantic roles" of the constituents in noun phrase queries. While some IM categories belong to named entities such as IM:*Director* for the intent class *Movie*, there can be semantic labels that are not named entities such as IH and IM:*Genre* (again for *Movie*).

3 Query Semantic Structure

Unlike database query languages such as SQL, web queries are usually formulated as sequences of words without explicit structures. This makes web queries difficult to interpret by computers. For example, should the query "aspirin side effect" be interpreted as "the side effect of aspirin" or "the aspirin of side effect"? Before trying to build models that can automatically makes such decisions, we first need to understand what constitute the semantic structure of a noun phrase query.

3.1 Definition

We let $\mathcal{C}$ denote a set of query intent classes that represent semantic concepts such as *Movie*, *Product* and *Drug*. The query constituents introduced below are all defined w.r.t. the intent class of a query, $c \in \mathcal{C}$, which is assumed to be known.

Intent head

An *intent head* (IH) is a query segment that corresponds to an **attribute name** of an intent class. For example, the IH of the query "alice in wonderland 2010 cast" is "cast", which is an attribute name of *Movie*. By issuing the query, the user intends to find out the values of the IH (*i.e.*, cast). A query can have multiple IHs, *e.g.*, "movie avatar director and cast". More importantly, there can be queries without an explicit IH. For example, "movie avatar" does not contain any segment that corresponds to an attribute name of *Movie*. Such a query, however, does have an implicit intent which is to obtain general information about the movie.

Intent modifier

In contrast, an *intent modifier* (IM) is a query segment that corresponds to an **attribute value** (of some attribute name). The role of IMs is to imposing constraints on the attributes of an intent class. For example, there are two constraints implied in the query "alice in wonderland 2010 cast": (1) the *Title* of the movie is "alice in wonderland"; and (2) the *Year* of the movie is "2010". Interestingly, the user does not explicitly specify the attribute names, *i.e.*, *Title* and *Year*, in this query. Such information, however, can be inferred given domain knowledge. In fact, one important goal of this work is to identify the semantic labels of IMs, *i.e.*, the attribute names they implicitly refer to. We use $\mathcal{A}_c$ to denote the set of IM semantic labels for the intent class c.

Other

Additionally, there can be query segments that do not play any semantic roles, which we refer to as Other.

3.2 Syntactic analysis

The notion of IHs and IMs in this work is closely related to that of linguistic *head nouns* and *modifiers* for noun phrases. In many cases, the IHs of noun phrase queries are exactly the head nouns in the linguistic sense. Exceptions mostly occur in queries without explicit IHs, *e.g.*, "movie avatar" in which the head noun "avatar" serves as an IM instead. Due to the strong resemblance, it is interesting to see if IHs can be identified by extracting linguistic head nouns from queries based on syntactic analysis. To this end, we apply the following heuristics for head noun extraction. We first run a POS-tagger and a chunker jointly on each query, where the POS-tagger/chunker is based on an HMM system trained on English Penn Treebank (Gao et al., 2001). We then mark the right most NP chunk before any prepositional phrase or adjective clause, and apply the NP head rules (Collins, 1999) to the marked NP chunk.

The main problem with this approach, however, is that a readily-available POS tagger or chunker is usually trained on natural language sentences and thus is unlikely to produce accurate results on web queries. As shown in (Barr et al., 2008), the lexical category distribution of web queries is dramatically different from that of natural languages. For example, prepositions and subordinating conjunctions, which are strong indicators of the syntactic

structure in natural languages, are often missing in web queries. Moreover, unlike most natural languages that follow the linear-order principle, web queries can have relatively free word orders (although some orders may occur more often than others statistically). These factors make it difficult to produce reliable syntactic analysis outputs. Consequently, the head nouns and hence the IHs extracted therefrom are likely to be error-prone, as will be shown by our experiments in Section 6.3.

Although a POS tagger and a chunker may not work well on queries, their output can be used as features for learning statistical models for semantic structure extraction, which we introduce next.

4 Models

This section presents two statistical models for semantic understanding of noun phrase queries. Assuming that the intent class $c \in \mathcal{C}$ of a query is known, we cast the problem of extracting the semantic structure of the query into a joint segmentation/classification problem. At a high level, we would like to identify query segments that correspond to IHs, IMs and Others. Furthermore, for each IM segment, we would like to assign a semantic label, denoted by IM:a, $a \in \mathcal{A}_c$, indicating which attribute name it refers to. In other words, our label set consists of $\mathcal{Y} = \{\text{IH}, \{\text{IM:}a\}_{a \in \mathcal{A}_c}, \text{Other}\}$.

Formally, we let $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ denote an input query of length M. To avoid confusion, we use i to represent the index of a word token and j to represent the index of a segment in the following text. Our goal is to obtain

$$\mathbf{s}^* = \operatorname*{argmax}_{\mathbf{s}} p(\mathbf{s}|c, \mathbf{x}) \tag{1}$$

where $\mathbf{s} = (s_1, s_2, \ldots, s_N)$ denotes a query segmentation as well as a classification of all segments. Each segment s_j is represented by a tuple (u_j, v_j, y_j). Here u_j and v_j are the indices of the starting and ending word tokens respectively; $y_j \in \mathcal{Y}$ is a label indicating the semantic role of s. We further augment the segment sequence with two special segments: *Start* and *End*, represented by s_0 and s_{N+1} respectively. For notional simplicity, we assume that the intent class is given and use $p(\mathbf{s}|\mathbf{x})$ as a shorthand for $p(\mathbf{s}|c, \mathbf{x})$, but keep in mind that the label space and hence the parameter space is class-dependent. Now we introduce two methods of modeling $p(\mathbf{s}|\mathbf{x})$.

4.1 CRFs

One natural approach to extracting the semantic structure of queries is to use linear-chain CRFs (Lafferty et al., 2001). They model the conditional probability of a label sequence given the input, where the labels, denoted as $\mathbf{y} = (y_1, y_2, \ldots, y_M)$, $y_i \in \mathcal{Y}$, have a one-to-one correspondence with the word tokens in the input.

Using linear-chain CRFs, we aim to find the label sequence that maximizes

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp\left\{\sum_{i=1}^{M+1} \lambda \cdot f(y_{i-1}, y_i, \mathbf{x}, i)\right\}. \tag{2}$$

The partition function $Z_\lambda(\mathbf{x})$ is a normalization factor. λ is a weight vector and $f(y_{i-1}, y_i, \mathbf{x})$ is a vector of feature functions referred to as a feature vector. The features used in CRFs will be described in Section 5.

Given manually-labeled queries, we estimate λ that maximizes the conditional likelihood of training data while regularizing model parameters. The learned model is then used to predict the label sequence $\mathbf{y}$ for future input sequences $\mathbf{x}$. To obtain $\mathbf{s}$ in Equation (1), we simply concatenate the maximum number of consecutive word tokens that have the same label and treat the resulting sequence as a segment. By doing this, we implicitly assume that there are no two adjacent segments with the same label in the true segment sequence. Although this assumption is not always correct in practice, we consider it a reasonable approximation given what we empirically observed in our training data.

4.2 Semi-Markov CRFs

In contrast to standard CRFs, semi-Markov CRFs directly model the segmentation of an input sequence as well as a classification of the segments (Sarawagi and Cohen, 2004), *i.e.*,

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \exp \sum_{j=1}^{N+1} \lambda \cdot f(s_{j-1}, s_j, \mathbf{x}) \tag{3}$$

In this case, the features $f(s_{j-1}, s_j, \mathbf{x})$ are defined on segments instead of on word tokens. More precisely, they are of the function form $f(y_{j-1}, y_j, \mathbf{x}, u_j, v_j)$. It is easy to see that by imposing a constraint $u_i = v_i$, the model is reduced to standard linear-chain CRFs. Semi-Markov CRFs make Markov assumptions at the segment level, thereby naturally offering means to

CRF features		
A1: Transition	$\delta(y_{i-1} = a)\delta(y_i = b)$	transiting from state a to b
A2: Lexical	$\delta(x_i = w)\delta(y_i = b)$	current word is w
A3: Semantic	$\delta(x_i \in \mathcal{W}_\mathcal{L})\delta(y_i = b)$	current word occurs in lexicon $\mathcal{L}$
A4: Semantic	$\delta(\mathbf{x}_{i-1:i} \in \mathcal{W}_\mathcal{L})\delta(y_i = b)$	current bigram occurs in lexicon $\mathcal{L}$
A5: Syntactic	$\delta(\text{POS}(\mathbf{x}_i) = z)\delta(y_i = b)$	POS tag of the current word is z
Semi-Markov CRF features		
B1: Transition	$\delta(y_{j-1} = a)\delta(y_j = b)$	Transiting from state a to b
B2: Lexical	$\delta(\mathbf{x}_{u_j:v_j} = \mathbf{w})\delta(y_j = b)$	Current segment is $\mathbf{w}$
B3: Lexical	$\delta(\mathbf{x}_{u_j:v_j} \ni w)\delta(y_j = b)$	Current segment contains word w
B4: Semantic	$\delta(\mathbf{x}_{u_j:v_j} \in \mathcal{L})\delta(y_j = b)$	Current segment is an element in lexicon $\mathcal{L}$
B5: Semantic	$\max_{l \in \mathcal{L}} s(\mathbf{x}_{u_j:v_j}, l)\delta(y_j = b)$	The max similarity between the segment and elements in $\mathcal{L}$
B6: Syntactic	$\delta(\text{POS}(\mathbf{x}_{u_j:v_j}) = \mathbf{z})\delta(y_j = b)$	Current segment's POS sequence is $\mathbf{z}$
B7: Syntactic	$\delta(\text{Chunk}(\mathbf{x}_{u_j:v_j}) = c)\delta(y_j = b)$	Current segment is a chunk with phrase type c

Table 2: A summary of feature types in CRFs and segmental CRFs for query understanding. We assume that the state label is b in all features and omit this in the feature descriptions.

incorporate segment-level features, as will be presented in Section 5.

5 Features

In this work, we explore the use of transition, lexical, semantic and syntactic features in Markov and semi-Markov CRFs. The mathematical expression of these features are summarized in Table 2 with details described as follows.

5.1 Transition features

Transition features, *i.e.*, A1 and B1 in Table 2, capture state transition patterns between adjacent word tokens in CRFs, and between adjacent segments in semi-Markov CRFs. We only use first-order transition features in this work.

5.2 Lexical features

In CRFs, a lexical feature (A2) is implemented as a binary function that indicates whether a specific word co-occurs with a state label. The set of words to be considered in this work are those observed in the training data. We can also generalize this type of features from words to n-grams. In other words, instead of inspecting the word identity at the current position, we inspect the n-gram identity by applying a window of length n centered at the current position.

Since feature functions are defined on segments in semi-Markov CRFs, we create B2 that indicates whether the phrase in a hypothesized query segment co-occurs with a state label. Here the set of phrase identities are extracted from the query segments in the training data. Furthermore, we create another type of lexical feature, B3, which is activated when a specific word occurs in a hypothesized query segment. The use of B3 would favor unseen words being included in adjacent segments rather than to be isolated as separate segments.

5.3 Semantic features

Models relying on lexical features may require very large amounts of training data to produce accurate prediction performance, as the feature space is in general large and sparse. To make our model generalize better, we create semantic features based on what we call *lexicons*. A lexicon, denoted as $\mathcal{L}$, is a cluster of semantically-related words/phrases. For example, a cluster of movie titles or director names can be such a lexicon. Before describing how such lexicons are generated for our task, we first introduce the forms of the semantic features assuming the availability of the lexicons.

We let $\mathcal{L}$ denote a lexicon, and $\mathcal{W}_\mathcal{L}$ denote the set of n-grams extracted from $\mathcal{L}$. For CRFs, we create a binary function that indicates whether *any* n-gram in $\mathcal{W}_\mathcal{L}$ co-occurs with a state label, with $n = 1, 2$ for A3, A4 respectively. For both A3 and A4, the number of such semantic features is equal to the number of lexicons multiplied by the number of state labels.

The same source of semantic knowledge can be conveniently incorporated in semi-Markov CRFs. One set of semantic features (B4) inspect whether the phrase of a hypothesized query segment matches *any* element in a given lexicon. A second set of semantic features (B5) relax the exact match constraints made by B4, and take as the feature value the maximum "similarity" between the query segment and *all* lexicon elements. The fol-

lowing similarity function is used in this work ,

$$s(\mathbf{x}_{u_j:v_j}, l) = 1 - \text{Lev}(\mathbf{x}_{u_j:v_j}, l)/|l| \quad (4)$$

where Lev represents the Levenshtein distance. Notice that we normalize the Levenshtein distance by the length of the lexicon element, as we empirically found it performing better compared with normalizing by the length of the segment. In computing the maximum similarity, we first retrieve a set of lexicon elements with a positive *tf-idf* cosine distance with the segment; we then evaluate Equation (4) for each retrieved element and find the one with the maximum similarity score.

Lexicon generation

To create the semantic features described above, we generate two types of lexicons leveraging databases and query logs for each intent class.

The first type of lexicon is an IH lexicon comprised of a list of attribute names for the intent class, *e.g.*, "box office" and "review" for the intent class *Movie*. One easy way of composing such a list is by aggregating the column names in the corresponding database such as Table 1. However, this approach may result in low coverage on IHs for some domains. Moreover, many database column names, such as *Title*, are unlikely to appear as IHs in queries. Inspired by Pasca and Van Durme (2007), we apply a bootstrapping algorithm that automatically learns attribute names for an intent class from query logs. The key difference from their work is that we create templates that consist of semantic labels at the segment level from training data. For example, "alice in wonderland 2010 cast" is labeled as "IM:*Title* IM:*Year* IH", and thus "IM:*Title* + IM:*Year* + #" is used as a template. We select the most frequent templates (top 2 in this work) from training data and use them to discover new IH phrases from the query log.

Secondly, we have a set IM lexicons, each comprised of a list of attribute values of an attribute name in $\mathcal{A}_c$. We exploit internal resources to generate such lexicons. For example, the lexicon for IM:*Title* (in *Movie*) is a list of movie titles generated by aggregating the values in the *Title* column of a movie database. Similarly, the lexicon for IM:*Employee* (in *Job*) is a list of employee names extracted from a job listing database. Note that a substantial amount of research effort has been dedicated to automatic lexicon acquisition from the Web (Pantel and Pennacchiotti, 2006; Pennacchiotti and Pantel, 2009). These techniques can be used in expanding the semantic lexicons for IMs when database resources are not available. But we do not use such techniques in our work since the lexicons extracted from databases in general have good precision and coverage.

5.4 Syntactic features

As mentioned in Section 3.2, web queries often lack syntactic cues and do not necessarily follow the linear order principle. Consequently, applying syntactic analysis such as POS tagging or chunking using models trained on natural language corpora is unlikely to give accurate results on web queries, as supported by our experimental evidence in Section 6.3. It may be beneficial, however, to use syntactic analysis results as additional evidence in learning.

To this end, we generate a sequence of POS tags for a given query, and use the co-occurrence of POS tag identities and state labels as syntactic features (A5) for CRFs.

For semi-Markov CRFs, we instead examine the POS tag sequence of the corresponding phrase in a query segment. Again their identities are combined with state labels to create syntactic features B6. Furthermore, since it is natural to incorporate segment-level features in semi-Markov CRFs, we can directly use the output of a syntactic chunker. To be precise, if a query segment is determined by the chunker to be a chunk, we use the indicator of the phrase type of the chunk (*e.g.*, NP, PP) combined with a state label as the feature, denoted by B7 in the Table. Such features are not activated if a query segment is determined not to be a chunk.

6 Evaluation

6.1 Data

To evaluate our proposed models and features, we collected queries from three domains, *Movie*, *Job* and *National Park*, and had them manually annotated. The annotation was given on both segmentation of the queries and classification of the segments according to the label sets defined in Table 3. There are 1000/496 samples in the training/test set for the *Movie* domain, 600/366 for the *Job* domain and 491/185 for the *National Park* domain. In evaluation, we report the test-set performance in each domain as well as the average performance (weighted by their respectively test-set size) over all domains.

Movie		Job		National Park	
IH	trailer, box office	IH	listing, salary	IH	lodging, calendar
IM:*Award*	oscar best picture	IM:*Category*	engineering	IM:*Category*	national forest
IM:*Cast*	johnny depp	IM:*City*	las vegas	IM:*City*	page
IM:*Character*	michael corleone	IM:*County*	orange	IM:*Country*	us
IM:*Category*	tv series	IM:*Employer*	walmart	IM:*Name*	yosemite
IM:*Country*	american	IM:*Level*	entry level	IM:*POI*	volcano
IM:*Director*	steven spielberg	IM:*Salary*	high-paying	IM:*Rating*	best
IM:*Genre*	action	IM:*State*	florida	IM:*State*	flordia
IM:*Rating*	best	IM:*Type*	full time		
IM:*Title*	the godfather				
Other	the, in, that	Other	the, in, that	Other	the, in, that

Table 3: Label sets and their respective query segment examples for the intent class *Movie*, *Job* and *National Park*.

6.2 Metrics

There are two evaluation metrics used in our work: segment F1 and sentence accuracy (Acc). The first metric is computed based on precision and recall at the segment level. Specifically, let us assume that the true segment sequence of a query is $\mathbf{s} = (s_1, s_2, \ldots, s_N)$, and the decoded segment sequence is $\mathbf{s}' = (s'_1, s'_2, \ldots, s'_K)$. We say that s'_k is a true positive if $s'_k \in \mathbf{s}$. The precision and recall, then, are measured as the total number of true positives divided by the total number of decoded and true segments respectively. We report the F1-measure which is computed as $2 \cdot \text{prec} \cdot \text{recall}/(\text{prec} + \text{recall})$.

Secondly, a sentence is correct if all decoded segments are true positives. Sentence accuracy is measured by the total number of correct sentences divided by the total number of sentences.

6.3 Results

We start with models that incorporate first-order transition features which are standard for both Markov and semi-Markov CRFs. We then experiment with lexical features, semantic features and syntactic features for both models. Table 4 and Table 5 give a summarization of all experimental results.

Lexical features

The first experiment we did is to evaluate the performance of lexical features (combined with transition features). This involves the use of A2 in Table 2 for CRFs, and B2 and B3 for semi-Markov CRFs. Note that adding B3, *i.e.,* indicators of whether a query segment contains a word identity, gave an absolute 7.0%/3.2% gain in sentence accuracy and segment F1 on average, as shown in the row B1-B3 in Table 5. For both A2 and B3, we also tried extending the features based on word IDs to those based on n-gram IDs, where $n = 1, 2, 3$. This greatly increased the number of lexical features but did not improve learning performance, most likely due to the limited amounts of training data coupled with the sparsity of such features. In general, lexical features do not generalize well to the test data, which accounts for the relatively poor performance of both models.

Semantic features

We created IM lexicons from three in-house databases on *Movie*, *Job* and *National Parks*. Some lexicons, *e.g.*, IM:*State*, are shared across domains. Regarding IH lexicons, we applied the bootstrapping algorithm described in Section 5.3 to a 1-month query log of *Bing*. We selected the most frequent 57 and 131 phrases to form the IH lexicons for *Movie* and *National Park* respectively. We do not have an IH lexicon for *Job* as the attribute names in that domain are much fewer and are well covered by training set examples.

We implemented A3 and A4 for CRFs, which are based on the n-gram sets created from lexicons; and B4 and B5 for semi-Markov CRFs, which are based on exact and fuzzy match with lexicon items. As shown in Table 4 and 5, drastic increases in sentence accuracies and F1-measures were observed for both models.

Syntactic features

As shown in the row A1-A5 in Table 4, combined with all other features, the syntactic features (A5) built upon POS tags boosted the CRF model performance. Table 6 listed the most dominant positive and negative features based on POS tags for *Movie* (features for the other two domains are not reported due to space limit). We can see that many of these features make intuitive sense. For

	Movie		Job		National Park		Average	
Features	Acc	F1	Acc	F1	Acc	F1	Acc	F1
A1,A2: Tran + Lex	59.9	75.8	65.6	84.7	61.6	75.6	62.1	78.9
A1-A3: Tran + Lex + Sem	67.9	80.2	70.8	87.4	70.5	80.8	69.4	82.8
A1-A4: Tran + Lex + Sem	72.4	83.5	72.4	89.7	71.1	82.3	72.2	85.0
A1-A5: Tran + Lex + Sem + Syn	74.4	84.8	75.1	89.4	75.1	85.4	74.8	86.5
A2-A5: Lex + Sem + Syn	64.9	78.8	68.1	81.1	64.8	83.7	65.4	81.0

Table 4: Sentence accuracy (Acc) and segment F1 (F1) using CRFs with different features.

	Movie		Job		National Park		Average	
Features	Acc	F1	Acc	F1	Acc	F1	Acc	F1
B1,B2: Tran + Lex	53.4	71.6	59.6	83.8	60.0	77.3	56.7	76.9
B1-B3: Tran + Lex	61.3	77.7	65.9	85.9	66.0	80.7	63.7	80.1
B1-B4: Tran + Lex + Sem	73.8	83.6	76.0	89.7	74.6	85.3	74.7	86.1
B1-B5: Tran + Lex + Sem	75.0	84.3	76.5	89.7	76.8	86.8	75.8	86.6
B1-B6: Tran + Lex + Sem + Syn	75.8	84.3	76.2	89.7	76.8	87.2	**76.1**	**86.7**
B1-B5,B7: Tran + Lex + Sem + Syn	75.6	84.1	76.0	89.3	76.8	86.8	75.9	86.4
B2-B6:Lex + Sem + Syn	72.0	82.0	73.2	87.9	76.5	89.3	73.8	85.6

Table 5: Sentence accuracy (Acc) and segment F1 (F1) using semi-Markov CRFs with different features.

example, IN (preposition or subordinating conjunction) is a strong indicator of Other, while TO and IM:*Date* usually do not co-occur. Some features, however, may appear less "correct". This is largely due to the inaccurate output of the POS tagger. For example, a large number of actor names were mis-tagged as RB, resulting in a high positive weight of the feature (RB, IM:*Cast*).

Positive	Negative
(IN, Other),	(TO, IM:*Date*)
(VBD, Other)	(IN, IM:*Cast*)
(CD, IM:*Date*)	(CD, IH)
(RB, IM:*Cast*)	(IN, IM:*Character*)

Table 6: Syntactic features with the largest positive/negative weights in the CRF model for *Movie*

Similarly, we added segment-level POS tag features (B6) to semi-Markov CRFs, which lead to the best overall results as shown by the highlighted numbers in Table 5. Again many of the dominant features are consistent with our intuition. For example, the most positive feature for *Movie* is (CD JJS, IM:*Rating*) (*e.g.* 100 best). When syntactic features based on chunking results (B7) are used instead of B6, the performance is not as good.

Transition features

In addition, it is interesting to see the importance of transition features in both models. Since web queries do not generally follow the linear order principle, is it helpful to incorporate transition features in learning? To answer this question, we dropped the transition features from the best systems, corresponding to the last rows in Table 4 and 5. This resulted in substantial degradations in performance. One intuitive explanation is that although web queries are relatively "order-free", statistically speaking, some orders are much more likely to occur than others. This makes it beneficial to use transition features.

Comparison to syntactic analysis

Finally, we conduct a simple experiment by using the heuristics described in Section 3.2 in extracting IHs from queries. The precision and recall of IHs averaged over all 3 domains are 50.4% and 32.8% respectively. The precision and recall numbers from our best model-based system, *i.e.*, B1-B6 in Table 5, are 89.9% and 84.6% respectively, which are significantly better than those based on pure syntactic analysis.

7 Conclusions

In this work, we make the first attempt to define the semantic structure of noun phrase queries. We propose statistical methods to automatically extract IHs, IMs and the semantic labels of IMs using a variety of features. Experiments show the effectiveness of semantic features and syntactic features in both Markov and semi-Markov CRF models. In the future, it would be useful to explore other approaches to automatic lexicon discovery to improve the quality or to increase the coverage of both IH and IM lexicons, and to systematically evaluate their impact on query understanding performance.

The author would like to thank Hisami Suzuki and Jianfeng Gao for useful discussions.

References

Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. 2009. Sources of evidence for vertical selection. In *SIGIR'09: Proceedings of the 32st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of English web-search queries. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1021–1030.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *ICML'05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jianfeng Gao, Jian-Yun Nie, Jian Zhang, Endong Xun, Ming Zhou, and Chang-Ning Huang. 2001. Improving query translation for CLIR using statistical models. In *SIGIR'01: Proceedings of the 24th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*.

Jinyoung Kim, Xiaobing Xue, and Bruce Croft. 2009. A probabilistic retrieval model for semistructured data. In *ECIR'09: Proceedings of the 31st European Conference on Information Retrieval*, pages 228–239.

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

Fangtao Li, Xian Zhang, Jinhui Yuan, and Xiaoyan Zhu. 2008a. Classifying what-type questions by head noun tagging. In *COLING'08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 481–488.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008b. Learning query intent from regularized click graph. In *SIGIR'08: Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, July.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR'09: Proceedings of the 32st Annual International ACM SIGIR conference on Research and Development in Information Retrieva*.

Mehdi Manshadi and Xiao Li. 2009. Semantic tagging of web search queries. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191.

Donald Metzler and Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. *Jounral of Information Retrieval*, 8(3).

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically har-vesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguis-tics and the 44th annual meeting of the ACL*, pages 113–120.

Stelios Paparizos, Alexandros Ntoulas, John Shafer, and Rakesh Agrawal. 2009. Answering web queries using structured data sources. In *Proceedings of the 35th SIGMOD international conference on Management of data*.

Marius Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI'07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Marius Pasca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08: HLT*.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *EMNLP'09: Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 238–247.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *CIKM'04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49.

Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems (NIPS'04)*.

Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *SIGIR'06: Proceedings of the 29th Annual International ACM SIGIR conference on research and development in information retrieval*, pages 131–138.

Multilingual Pseudo-Relevance Feedback: Performance Study of Assisting Languages

Manoj K. Chinnakotla Karthik Raman Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay,
Mumbai, India
{manoj,karthikr,pb}@cse.iitb.ac.in

Abstract

In a previous work of ours Chinnakotla et al. (2010) we introduced a novel framework for Pseudo-Relevance Feedback (PRF) called *MultiPRF*. Given a query in one language called *Source*, we used English as the *Assisting Language* to improve the performance of PRF for the source language. MulitiPRF showed remarkable improvement over plain Model Based Feedback (MBF) uniformly for 4 languages, *viz., French, German, Hungarian* and *Finnish* with *English* as the assisting language. This fact inspired us to study the effect of *any source-assistant* pair on MultiPRF performance from out of a set of languages with widely different characteristics, *viz., Dutch, English, Finnish, French, German* and *Spanish*. Carrying this further, we looked into the effect of using *two assisting languages together* on PRF.

The present paper is a report of these investigations, their results and conclusions drawn therefrom. While performance improvement on MultiPRF is observed whatever the assisting language and whatever the source, observations are mixed when two assisting languages are used simultaneously. Interestingly, the performance improvement is more pronounced when the source and assisting languages are *closely related*, e.g., French and Spanish.

1 Introduction

The central problem of Information Retrieval (IR) is to satisfy the user's information need, which is typically expressed through a short (typically 2-3 words) and often ambiguous query. The problem of matching the user's query to the documents is rendered difficult by natural language phenomena like *morphological variations*, *polysemy* and *synonymy*. Relevance Feedback (RF) tries to overcome these problems by eliciting user feedback on the relevance of documents obtained from the initial ranking and then uses it to automatically refine the query. Since user input is hard to obtain, Pseudo-Relevance Feedback (PRF) (Buckley et al., 1994; Xu and Croft, 2000; Mitra et al., 1998) is used as an alternative, wherein RF is performed by *assuming* the top k documents from the initial retrieval as being *relevant* to the query. Based on the above assumption, the terms in the feedback document set are analyzed to choose the most distinguishing set of terms that characterize the feedback documents and as a result the relevance of a document. Query refinement is done by adding the terms obtained through PRF, along with their weights, to the actual query.

Although PRF has been shown to improve retrieval, it suffers from the following drawbacks: (a) the type of term associations obtained for query expansion is restricted to co-occurrence based relationships in the feedback documents, and thus other types of term associations such as lexical and semantic relations (morphological variants, synonyms) are not explicitly captured, and (b) due to the inherent assumption in PRF, *i.e.*, relevance of top k documents, performance is sensitive to that of the initial retrieval algorithm and as a result is not robust.

Multilingual Pseudo-Relevance Feedback (MultiPRF) (Chinnakotla et al., 2010) is a novel framework for PRF to overcome both the above limitations of PRF. It does so by taking the help of a different language called the *assisting language*. In MultiPRF, given a query in source language L_1, the query is automatically translated into the assisting language L_2 and PRF performed in the assisting language. The resultant terms are translated back into L_1 using a probabilistic bi-lingual dictionary. The translated feedback

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1346–1356,
Uppsala, Sweden, 11-16 July 2010.

model, is then combined with the original feedback model of L_1 to obtain the final model which is used to re-rank the corpus. MulitiPRF showed remarkable improvement on standard CLEF collections over plain Model Based Feedback (MBF) uniformly for 4 languages, *viz., French, German, Hungarian* and *Finnish* with *English* as the assisting language. This fact inspired us to study the effect of *any source-assistant* pair on PRF performance from out of a set of languages with widely different characteristics, *viz., Dutch, English, Finnish, French, German* and *Spanish*. Carrying this further, we looked into the effect of using *two assisting languages together* on PRF.

The present paper is a report of these investigations, their results and conclusions drawn therefrom. While performance improvement on PRF is observed whatever the assisting language and whatever the source, observations are mixed when two assisting languages are used simultaneously. Interestingly, the performance improvement is more pronounced when the source and assisting languages are *closely related*, e.g., French and Spanish.

The paper is organized as follows: Section 2, discusses the related work. Section 3, explains the Language Modeling (LM) based PRF approach. Section 4, describes the MultiPRF approach. Section 5 discusses the experimental set up. Section 6 presents the results, and studies the effect of varying the assisting language and incorporates multiple assisting languages. Finally, Section 7 concludes the paper by summarizing and outlining future work.

2 Related Work

PRF has been successfully applied in various IR frameworks like vector space models, probabilistic IR and language modeling (Buckley et al., 1994; Jones et al., 2000; Lavrenko and Croft, 2001; Zhai and Lafferty, 2001). Several approaches have been proposed to improve the performance and robustness of PRF. Some of the representative techniques are (i) Refining the feedback document set (Mitra et al., 1998; Sakai et al., 2005), (ii) Refining the terms obtained through PRF by selecting good expansion terms (Cao et al., 2008) and (iii) Using selective query expansion (Amati et al., 2004; Cronen-Townsend et al., 2004) and (iv) Varying the importance of documents in the feedback set (Tao and Zhai, 2006). Another direction of work, often reported in the TREC Robust Track, is to use a large external collection like Wikipedia or the Web as a source of expansion terms (Xu et al., 2009; Voorhees, 2006). The intuition behind the above approach is that if the query does not have many relevant documents in the collection then any improvements in the modeling of PRF is bound to perform poorly due to query drift.

Several approaches have been proposed for including different types of lexically and semantically related terms during query expansion. Voorhees (1994) use Wordnet for query expansion and report negative results. Recently, random walk models (Lafferty and Zhai, 2001; Collins-Thompson and Callan, 2005) have been used to learn a rich set of term level associations by combining evidence from various kinds of information sources like WordNet, Web *etc*. Metzler and Croft (2007) propose a feature based approach called *latent concept expansion* to model term dependencies.

All the above mentioned approaches use the resources available *within* the language to improve the performance of PRF. However, we make use of a *second language* to improve the performance of PRF. Our proposed approach is especially attractive in the case of resource-constrained languages where the original retrieval is bad due to poor coverage of the collection and/or inherent complexity of query processing (for example *term conflation*) in those languages.

Jourlin et al. (1999) use parallel blind relevance feedback, *i.e.* they use blind relevance feedback on a larger, more reliable parallel corpus, to improve retrieval performance on imperfect transcriptions of speech. Another related idea is by Xu et al. (2002), where a statistical thesaurus is learned using the probabilistic bilingual dictionaries of Arabic to English and English to Arabic. Meij et al. (2009) tries to expand a query in a different language using language models for domain-specific retrieval, but in a very different setting. Since our method uses a corpus in the assisting language from a similar time period, it can be likened to the work by Talvensaari et al. (2007) who used comparable corpora for Cross-Lingual Information Retrieval (CLIR). Other work pertaining to document alignment in comparable corpora, such as Braschler and Schäuble (1998), Lavrenko et al. (2002), also share certain common themes with our approach. Recent work by Gao et al.

(2008) uses English to improve the performance over a subset of Chinese queries whose translations in English are unambiguous. They use inter-document similarities across languages to improve the ranking performance. However, cross language document similarity measurement is in itself known to be an hard problem and the scale of their experimentation is quite small.

3 PRF in the LM Framework

The Language Modeling (LM) Framework allows PRF to be modelled in a principled manner. In the LM approach, documents and queries are modeled using multinomial distribution over words called *document language model* $P(w|D)$ and *query language model* $P(w|\Theta_Q)$ respectively. For a given query, the document language models are ranked based on their proximity to the query language model, measured using KL-Divergence.

$$KL(\Theta_Q||D) = \sum_w P(w|\Theta_Q) \cdot log \frac{P(w|\Theta_Q)}{P(w|D)}$$

Since the query length is short, it is difficult to estimate Θ_Q accurately using the query alone. In PRF, the top *k* documents obtained through the initial ranking algorithm are assumed to be relevant and used as feedback for improving the estimation of Θ_Q. The feedback documents contain both relevant and noisy terms from which the feedback language model is inferred based on a Generative Mixture Model (Zhai and Lafferty, 2001).

Let $D_F = \{d_1, d_2, \ldots, d_k\}$ be the top k documents retrieved using the initial ranking algorithm. Zhai and Lafferty (Zhai and Lafferty, 2001) model the feedback document set D_F as a mixture of two distributions: (a) the *feedback language model* and (b) the *collection model* $P(w|C)$. The feedback language model is inferred using the EM Algorithm (Dempster et al., 1977), which iteratively accumulates probability mass on the most *distinguishing* terms, *i.e.* terms which are more frequent in the feedback document set than in the entire collection. To maintain query focus the final converged feedback model, Θ_F is interpolated with the initial query model Θ_Q to obtain the final query model Θ_{Final}.

$$\Theta_{Final} = (1 - \alpha) \cdot \Theta_Q + \alpha \cdot \Theta_F$$

Θ_{Final} is used to re-rank the corpus using the KL-Divergence ranking function to obtain the final ranked list of documents. Henceforth, we refer to the above technique as *Model Based Feedback (MBF)*.

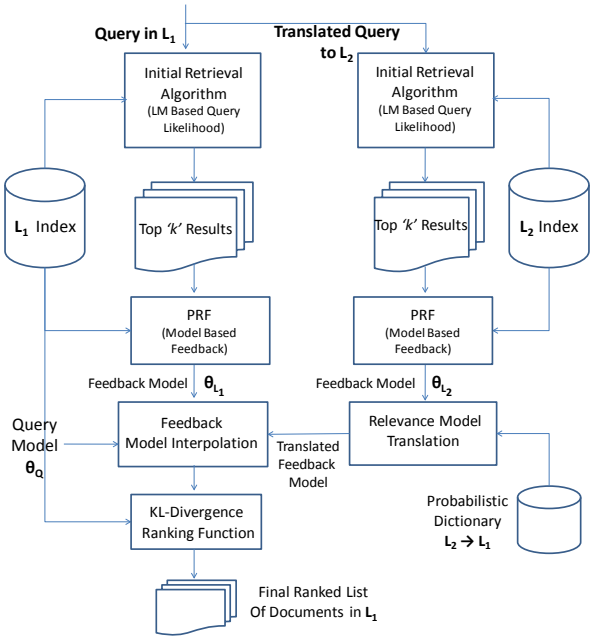


Figure 1: Schematic of the Multilingual PRF Approach

Symbol	Description
Θ_Q	Query Language Model
$\Theta^F_{L_1}$	Feedback Language Model obtained from PRF in L_1
$\Theta^F_{L_2}$	Feedback Language Model obtained from PRF in L_2
$\Theta^{Trans}_{L_1}$	Feedback Model Translated from L_2 to L_1
$t(f\|e)$	Probabilistic Bi-Lingual Dictionary from L_2 to L_1
β, γ	Interpolation coefficients coefficients used in MultiPRF

Table 2: Glossary of Symbols used in explaining MultiPRF

4 Multilingual PRF (MultiPRF)

The schematic of the MultiPRF approach is shown in Figure 1. Given a query Q in the source language L_1, we automatically translate the query into the assisting language L_2. We then rank the documents in the L_2 collection using the query likelihood ranking function (John Lafferty and Chengxiang Zhai, 2003). Using the top k documents, we estimate the feedback model using MBF as described in the previous section. Similarly, we also estimate a feedback model using the original query and the top k documents retrieved from the initial ranking in L_1. Let the resultant feedback models be $\Theta^F_{L_2}$ and $\Theta^F_{L_1}$ respectively. The feedback model estimated in the assisting language $\Theta^F_{L_2}$ is translated back into language L_1 using a probabilistic bi-lingual dictionary $t(f|e)$ from $L_2 \rightarrow L_1$ as follows:

$$P(f|\Theta^{Trans}_{L_1}) = \sum_{\forall\ e \text{ in } L_2} t(f|e) \cdot P(e|\Theta^F_{L_2}) \quad (1)$$

The probabilistic bi-lingual dictionary $t(f|e)$ is

Language	CLEF Collection Identifier	Description	No. of Documents	No. of Unique Terms	CLEF Topics (No. of Topics)
English	EN-00+01+02	LA Times 94	113005	174669	-
	EN-03+05+06	LA Times 94, Glasgow Herald 95	169477	234083	-
	EN-02+03	LA Times 94, Glasgow Herald 95	169477	234083	91-200 (67)
French	FR-00	Le Monde 94	44013	127065	1-40 (29)
	FR-01+02	Le Monde 94, French SDA 94	87191	159809	41-140 (88)
	FR-02+03	Le Monde 94, French SDA 94-95	129806	182214	91-200 (67)
	FR-03+05	Le Monde 94, French SDA 94-95	129806	182214	141-200,251-300 (99)
	FR-06	Le Monde 94-95, French SDA 94-95	177452	231429	301-350 (48)
German	DE-00	Frankfurter Rundschau 94, Der Spiegel 94-95	153694	791093	1-40 (33)
	DE-01+02	Frankfurter Rundschau 94, Der Spiegel 94-95, German SDA 94	225371	782304	41-140 (85)
	DE-02+03	Frankfurter Rundschau 94, Der Spiegel 94-95, German SDA 94-95	294809	867072	91-200 (67)
	DE-03	Frankfurter Rundschau 94, Der Spiegel 94-95, German SDA 94-95	294809	867072	141-200 (51)
Finnish	FI-02+03+04	Aamulehti 94-95	55344	531160	91-250 (119)
	FI-02+03	Aamulehti 94-95	55344	531160	91-200 (67)
Dutch	NL-02+03	NRC Handelsblad 94-95, Algemeen Dagblad 94-95	190604	575582	91-200 (67)
Spanish	ES-02+03	EFE 94, EFE 95	454045	340250	91-200 (67)

Table 1: Details of the CLEF Datasets used for Evaluating the MultiPRF approach. The number shown in brackets of the final column CLEF Topics indicate the actual number of topics used during evaluation.

Source Term	Top Aligned Terms in Target
French	**English**
américain	american, us, united, state, america
nation	nation, un, united, state, country
eťude	study, research, assess, investigate, survey
German	**English**
flugzeug	aircraft, plane, aeroplane, air, flight
spiele	play, game, stake, role, player
verhältnis	relationship, relate, balance, proportion

Table 3: Top Translation Alternatives for some sample words in Probabilistic Bi-Lingual Dictionary

learned from a parallel sentence-aligned corpora in $L_1 - L_2$ based on word level alignments. Tiedemann (Tiedemann, 2001) has shown that the translation alternatives found using word alignments could be used to infer various morphological and semantic relations between terms. In Table 3, we show the top translation alternatives for some sample words. For example, the French word *américain* (american) brings different variants of the translation like *american, america, us, united, state, america* which are lexically and semantically related. Hence, the probabilistic bi-lingual dictionary acts as a rich source of morphologically and semantically related feedback terms. Thus, during this step, of translating the feedback model as given in Equation 1, the translation model adds related terms in L_1 which have their source as the term from feedback model $\Theta^F_{L_2}$. The final MultiPRF model is obtained by interpolating the above translated feedback model with the original query model and the feedback model of language L_1 as given below:

$$\Theta^{Multi}_{L_1} = (1 - \beta - \gamma) \cdot \Theta_Q + \beta \cdot \Theta^F_{L_1} + \gamma \cdot \Theta^{Trans}_{L_1} \quad (2)$$

Since we want to retain the query focus during back translation the feedback model in L_2 is interpolated with the translated query before translation of the L_2 feedback model. The parameters β and γ control the relative importance of the original query model, feedback model of L_1 and the translated feedback model obtained from L_1 and are tuned based on the choice of L_1 and L_2.

5 Experimental Setup

We evaluate the performance of our system using the standard CLEF evaluation data in six languages, widely varying in their familial relationships - Dutch, German, English, French, Spanish and Finnish using more than 600 topics. The details of the collections and their corresponding topics used for MultiPRF are given in Table 1. Note that, in each experiment, we choose assisting collections such that the topics in the source language are covered in the assisting collection so as to get meaningful feedback terms. In all the topics, we only use the *title* field. We ignore the topics which have no relevant documents as the true performance on those topics cannot be evaluated.

We demonstrate the performance of MultiPRF approach with French, German and Finnish as source languages and Dutch, English and Spanish as the assisting language. We later vary the assisting language, for each source language and study the effects. We use the Terrier IR platform (Ounis et al., 2005) for indexing the documents. We perform standard tokenization, stop word removal and stemming. We use the Porter Stemmer for English and the stemmers available through the Snowball package for other languages. Other than these, we do not perform any language-specific processing on the languages. In case of French,

Collection	Assist. Lang	P@5			P@10			MAP			GMAP		
		MBF	MultiPRF	% Impr.	MBF	MultiPRF	% Impr.	MBF	MultiPRF	% Impr.	MBF	MultiPRF	% Impr.
FR-00	EN	0.4690	0.5241	**11.76‡**	0.4000	0.4000	0.00	0.4220	0.4393	4.10	0.2961	0.3413	**15.27**
	ES		0.5034	**7.35‡**		0.4103	2.59		0.4418	4.69		0.3382	**14.22**
	NL		0.5034	7.35		0.4103	2.59		0.4451	5.47		0.3445	**16.34**
FR-01+02	EN	0.4636	0.4818	3.92	0.4068	0.4386	**7.82‡**	0.4342	0.4535	**4.43‡**	0.2395	0.2721	**13.61**
	ES		0.4977	**7.35‡**		0.4363	**7.26‡**		0.4416	1.70		0.2349	-1.92
	NL		0.4818	3.92		0.4409	**8.38‡**		0.4375	0.76		0.2534	**5.80**
FR-03+05	EN	0.4545	0.4768	**4.89‡**	0.4040	0.4202	**4‡**	0.3529	0.3694	**4.67‡**	0.1324	0.1411	**6.57**
	ES		0.4727	4.00		0.4080	1.00		0.3582	1.50		0.1325	0.07
	NL		0.4525	-0.44		0.4010	-0.75		0.3513	0.45		0.1319	-0.38
FR-06	EN	0.4917	0.5083	3.39	0.4625	0.4729	2.25	0.3837	0.4104	6.97	0.2174	0.2810	**29.25**
	ES		0.5083	3.39		0.4687	1.35		0.3918	2.12		0.2617	**20.38**
	NL		0.5083	3.39		0.4646	0.45		0.3864	0.71		0.2266	**4.23**
DE-00	EN	0.2303	0.3212	**39.47‡**	0.2394	0.2939	**22.78‡**	0.2158	0.2273	5.31	0.0023	0.0191	**730.43**
	ES		0.3212	**39.47‡**		0.2818	**17.71‡**		0.2376	10.09		0.0123	**434.78**
	NL		0.3151	**36.82‡**		0.2818	**17.71‡**		0.2331	8.00		0.0122	**430.43**
DE-01+02	EN	0.5341	0.6000	**12.34‡**	0.4864	0.5318	**9.35‡**	0.4229	0.4576	**8.2‡**	0.1765	0.2721	**9.19**
	ES		0.5682	**6.39‡**		0.5091	**4.67‡**		0.4459	5.43		0.2309	**30.82**
	NL		0.5773	**8.09‡**		0.5114	**5.15‡**		0.4498	**6.35‡**		0.2355	**33.43**
DE-03	EN	0.5098	0.5412	6.15	0.4784	0.4980	4.10	0.4274	0.4355	1.91	0.1243	0.1771	**42.48**
	ES		0.5647	**10.77‡**		0.4980	4.10		0.4568	**6.89‡**		0.1645	**32.34**
	NL		0.5529	**8.45‡**		0.4941	3.27		0.4347	1.72		0.1490	**19.87**
FI-02+03+04	EN	0.3782	0.4034	**6.67‡**	0.3059	0.3319	**8.52‡**	0.3966	0.4246	**7.06‡**	0.1344	0.2272	**69.05**
	ES		0.3879	2.58		0.3267	6.81		0.3881	-2.15		0.1755	**30.58**
	NL		0.3948	4.40		0.3301	7.92		0.4077	2.79		0.1839	**36.83**

Table 4: Results comparing the performance of MultiPRF over baseline MBF on CLEF collections with English (EN), Spanish (ES) and Dutch (NL) as assisting languages. Results marked as ‡ indicate that the improvement was found to be statistically significant over the baseline at 90% confidence level ($\alpha = 0.01$) when tested using a paired two-tailed t-test.

since some function words like *l'*, *d' etc.,* occur as prefixes to a word, we strip them off during indexing and query processing, since it significantly improves the baseline performance. We use standard evaluation measures like *MAP*, *P@5* and *P@10* for evaluation. Additionally, for assessing robustness, we use the Geometric Mean Average Precision (GMAP) metric (Robertson, 2006) which is also used in the TREC Robust Track (Voorhees, 2006). The probabilistic bi-lingual dictionary used in MultiPRF was learnt automatically by running GIZA++: a word alignment tool (Och and Ney, 2003) on a parallel sentence aligned corpora. For all the above language pairs we used the *Europarl Corpus* (Philipp, 2005). We use Google Translate as the query translation system as it has been shown to perform well for the task (Wu et al., 2008). We use the MBF approach explained in Section 3 as a baseline for comparison. We use two-stage Dirichlet smoothing with the optimal parameters tuned based on the collection (Zhai and Lafferty, 2004). We tune the parameters of MBF, specifically λ and α, and choose the values which give the optimal performance on a given collection. We uniformly choose the top ten documents for feedback. Table 4 gives the overall results.

6 Results and Discussion

In Table 4, we see the performance of the MultiPRF approach for three assisting languages, and how it compares with the baseline MBF methods. We find MultiPRF to consistently outperform the baseline value on all metrics, namely MAP (where significant improvements range from 4.4% to 7.1%); P@5 (significant improvements range from 4.9% to 39.5% and P@10 (where MultiPRF has significant gains varying from 4% to 22.8%). Additionally we also find MultiPRF to be more robust than the baseline, as indicated by the GMAP score, where improvements vary from 4.2% to 730%. Furthermore we notice these trends hold across different assisting languages, with Spanish and Dutch outperforming English as the assisting language on some of the French and German collections. On performing a more detailed study of the results we identify the main reason for improvements in our approach is the ability to obtain good feedback terms in the assisting language coupled with the introduction of lexically and semantically related terms during the back-translation step.

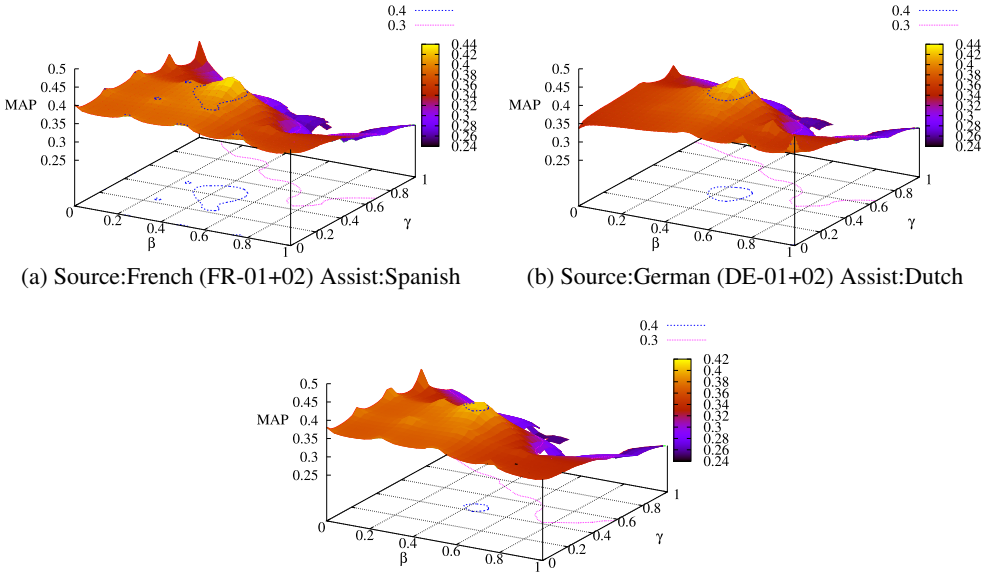
In Table 5, we see some examples, which illustrates the feedback terms brought by the MultiPRF method. As can be seen by these example, the gains achieved by MultiPRF are primarily due to one of three reasons: (a) Good Feedback in Assisting Language: If the feedback model in the assisting language contains good terms, then the back-translation process will introduce the corresponding feedback terms in the source language, thus leading to improved performance. As an example of this phenomena, consider the French Query *"Maladie de Creutzfeldt-Jakob"*. In this case the original feedback model also performs

TOPIC NO	ASSIST LANG.	SOURCE LANGUAGE QUERY	TRANSLATED QUERY	QUERY MEANING	MBF MAP	MPRF MAP	MBF- Top Representative Terms (With Meaning) Excl. Query Terms	MultiPRF- Top Representative Terms (With Meaning) Excl. Query Terms
GERMAN '01: TOPIC 61	EN	Ölkatastrophe in Sibirien	Oil Spill in Siberia	Siberian Oil Catastrophe	0.618	0.812	exxon, million, ol (oil), tonn, russisch (russian), olp (oil), moskau (moscow), us	olverschmutz (oil pollution), ol, russisch, erdol (petroleum), russland (russia), olunfall(oil spill), olp
GERMAN '02: TOPIC 105	ES	Bronchialasthma	El asma bronquial	Bronchial Asthma	0.062	0.636	chronisch (chronic), pet, athlet (athlete), ekrank (ill), gesund (healthy), tuberkulos (tuberculosis), patient, reis (rice), person	asthma, allergi, krankheit (disease), allerg (allergenic), chronisch, hauterkrank (illness of skin), arzt (doctor), erkrank (ill)
FRENCH '02: TOPIC 107	NL	Ingénierie génétique	Genetische Manipulatie	Genetic Engineering	0.145	0.357	dévelopg (developed), évolu (evolved), product, produit (product), moléculair (molecular)	genetic, gen, engineering, dévelopg, product
FRENCH '06: TOPIC 256	EN	Maladie de Creutzfeldt-Jakob	Creutzfeldt-Jakob	Creutzfeldt-Jakob Disease	0.507	0.688	malad (illness), produit (product), animal (animal), hormon (hormone)	malad, humain (human), bovin (bovine), encéphalopath (suffering from encephalitis), scientif, recherch (research)
GERMAN '03: TOPIC 157	EN	Siegerinnen von Wimbledon	Champions of Wimbledon	Wimbledon Lady Winners	0.074	0.146	telefonbuch (phone book), sieg (victory), titelseit (front page), telekom (telecommunication), graf	gross (large), verfecht (champion), sampra (sampras), 6, champion, steffi, verteidigt (defendending), martina, jovotna, navratilova
GERMAN '01: TOPIC 91	ES	AI in Lateinamerika	La gripe aviar en América Latina	AI in Latin America	0.456	0.098	international, amnesty, strassenkind (street child), kolumbi (Columbian), land, brasili (Brazil), menschenrecht (human rights), polizei (police)	karib (Caribbean), land, brasili, schuld (blame), amerika, kalt (cold), welt (world), forschung (research)
GERMAN '03: TOPIC 196	EN	Fusion japanischer Banken	Fusion of Japanese banks	Merger of Japanese Banks	0.572	0.264	daiwa, tokyo, filial (branch), zusammenschluss (merger)	kernfusion (nuclear fusion), zentralbank (central bank), daiwa, weltbank (world bank), investitionsbank (investment bank)
FRENCH '03: TOPIC 152	NL	Les droits de l'enfant	De rechten van het kind	Child Rights	0.479	0.284	convent (convention), franc, international, onun (united nations), réserv (reserve)	per (father), convent, franc, jurid (legal), homm (man), cour (court), biolog

Table 5: Qualitative comparison of feedback terms given by MultiPRF and MBF on representative queries where positive and negative results were observed in French and German collections.

quite strongly with a MAP score of 0.507. Although there is no significant topic drift in this case, there are not many relevant terms apart from the query terms. However the same query performs very well in English with all the documents in the feedback set of the English corpus being relevant, thus resulting in informative feedback terms such as {*bovin, scientif, recherch*}. (b) Finding Synonyms/Morphological Variations: Another situation in which MultiPRF leads to large improvements is when it finds semantically/lexically related terms to the query terms which the original feedback model was unable to. For example, consider the French query *"Ingénierie g´n´tique"*. While the feedback model was unable to find any of the synonyms of the query terms, due to their lack of co-occurence with the query terms, the MultiPRF model was able to get these terms, which are introduced primarily during the back-translation process. Thus terms like {*genetic, gen, engineering*}, which are synonyms of the query words, are found thus resulting in improved performance. (c) Combination of Above Factors: Sometimes a combination of the above two factors causes improvements in the performance as in the German query *"Ölkatastrophein Sibirien"*. For this query, MultiPRF finds good feedback terms such as {*russisch, russland*} while also obtaining semantically related terms such as {*olverschmutz, erdol, olunfall*}.

Although all of the previously described examples had good quality translations of the query in the assisting language, as mentioned in (Chinnakotla et al., 2010), the MultiPRF approach is robust to suboptimal translation quality as well. To see how MultiPRF leads to improvements even with errors in query translation consider the German Query *"Siegerinnen von Wimbledon"*. When this is translated to English, the term "Lady" is dropped, this causes only "Wimbledon Champions" to remain. As can be observed, this causes terms like *sampras* to come up in the MultiPRF model. However, while the MultiPRF model has some terms pertaining to Men's Winners of Wimbledon as well, the original feedback model suffers from severe topic drift, with irrelevant terms such as {*telefonbuch, telekom*} also amongst the top terms. Thus we notice that despite the error in query translation MultiPRF still manages to correct the drift of the original feedback model, while also introducing relevant terms such as {*verfecht, steffi, martina, novotna, navratilova*} as well. Thus as shown in (Chinnakotla et al., 2010), having a better query translation system can only lead to better performance. We also perform a detailed error analysis and found three main reasons for MultiPRF failing: (i) Inaccuracies in query translation (including the presence of out-of-vocabulary terms). This is seen in the German Query *AI in Lateinamerika*, which wrongly translates to *Avian Flu in Latin America* in Spanish thus affecting performance. (ii) Poor retrieval in Assisting Language. Consider the French query *Les droits de l'enfant*, for which due to topic drift in English, MultiPRF performance reduces. (iii) In a few rare cases inaccuracy in the back transla-



(a) Source:French (FR-01+02) Assist:Spanish

(b) Source:German (DE-01+02) Assist:Dutch

(c) Source:Finnish (FI-02+03+04) Assist:English

Figure 2: Results showing the sensitivity of MultiPRF performance to parameters β and γ for French, German and Finnish.

tion affects performance as well.

6.1 Parameter Sensitivity Analysis

The MultiPRF parameters β and γ in Equation 2 control the relative importance assigned to the original feedback model in source language L_1, the translated feedback model obtained from assisting language L_2 and the original query terms. We varied the β and γ parameters for French, German and Finnish collections with English, Dutch and Spanish as assisting languages and studied its effect on MAP of MultiPRF. The results are shown in Figure 2. The results show that, in all the three collections, the optimal value of the parameters almost remains the same and lies in the range of 0.4-0.48. Due to the above reason, we arbitrarily choose the parameters in the above range and do not use any technique to learn these parameters.

6.2 Effect of Assisting Language Choice

In this section, we discuss the effect of varying the assisting language. Besides, we also study the inter and intra familial behaviour of source-assisting language pairs. In order to ensure that the results are comparable across languages, we indexed the collections from the years 2002, 2003 and use common topics from the topic range 91-200 that have relevant documents across all the six languages. The number of such common topics were 67. For each source language, we use the other languages as assisting collections and study the performance of MultiPRF. Since query translation quality varies across language pairs, we analyze the behaviour of MultiPRF in the following two scenarios: (a) Using ideal query translation (b) Using Google Translate for query translation. In ideal query translation setup, in order to eliminate its effect, we skip the query translation step and use the corresponding original topics for each target language instead. The results for both the above scenarios are given in Tables 6 and 7.

From the results, we firstly observe that besides English, other languages such as French, Spanish, German and Dutch act as good assisting languages and help in improving performance over monolingual MBF. We also observe that the best assisting language varies with the source language. However, the crucial factors of the assisting language which influence the performance of MultiPRF are: (a) *Monolingual PRF Performance:* The main motivation for using a different language was to get good feedback terms, especially in case of queries which fail in the source language. Hence, an assisting language in which the monolingual feedback performance itself is poor, is unlikely to give any performance gains. This observation is evident in case of Finnish, which has the lowest Monolingual MBF performance. The results show that Finnish is the least helpful of assisting languages, with performance similar to those of the baselines. We also observe that the three best performing assistant languages, i.e. English, French and Spanish, have the highest monolingual performances as well, thus further validating the claim. One possible reason for this is the relative

Source Lang.		Assisting Language						Source Lang.MBF
		English	German	Dutch	Spanish	French	Finnish	
English	MAP	-	0.4464 (-0.7%)	0.4471 (-0.5%)	0.4566 (+1.6%)	0.4563 (+1.5%)	0.4545 (+1.1%)	0.4495
	P@5		0.4925 (-0.6%)	0.5045 (+1.8%)	0.5164 (+4.2%)	0.5075 (+2.4%)	**0.5194 (+4.8%)**	0.4955
	P@10		0.4343 (+0.4%)	0.4373 (+1.0%)	**0.4537 (+4.8%)**	0.4343 (+0.4%)	0.4373 (+1.0%)	0.4328
German	MAP	**0.4229 (+4.9%)**	-	**0.4346 (+7.8%)**	**0.4314 (+7.0%)**	0.411 (+1.9%)	0.3863 (-4.2%)	0.4033
	P@5	**0.5851 (+14%)**		**0.5851 (+14%)**	**0.5791 (+12.8%)**	**0.594 (+15.7%)**	**0.5522 (+7.6%)**	0.5134
	P@10	**0.5284 (+11.3%)**		**0.5209 (+9.8%)**	**0.5179 (+9.1%)**	**0.5149 (+8.5%)**	**0.5075 (+6.9%)**	0.4746
Dutch	MAP	0.4317 (+4%)	**0.4453 (+7.2%)**	-	0.4275 (+2.9%)	0.4241 (+2.1%)	0.3971 (-4.4%)	0.4153
	P@5	**0.5642 (+11.8%)**	**0.5731 (+13.6%)**		**0.5343 (+5.9%)**	**0.5582 (+10.6%)**	0.5045 (0%)	0.5045
	P@10	**0.5075 (+9%)**	**0.4925 (+5.8%)**		**0.4896 (+5.1%)**	**0.5015 (+7.7%)**	0.4806 (+3.2%)	0.4657
Spanish	MAP	0.4667 (-2.9%)	0.4749 (-1.2%)	0.4744 (-1.3%)	-	0.4609 (-4.1%)	0.4311 (-10.3%)	0.4805
	P@5	0.62 (-2.9%)	0.6418 (+0.5%)	0.6299 (-1.4%)		0.6269 (-1.6%)	0.6149 (-3.7%)	0.6388
	P@10	0.5625 (-1.8%)	0.5806 (+1.3%)	0.5851 (+2.1%)		0.5627 (-1.8%)	0.5478 (-4.4%)	0.5731
French	MAP	**0.4658 (+6.9%)**	0.4526 (+3.9%)	0.4374 (+0.4%)	**0.4634 (+6.4%)**	-	0.4451 (+2.2%)	0.4356
	P@5	0.4925 (+3.1%)	0.4806 (+0.6%)	0.4567 (-4.4%)	0.4925 (+3.1%)		0.4836 (+1.3%)	0.4776
	P@10	0.4358 (+3.9%)	0.4239 (+1%)	0.4224 (+0.7%)	**0.4388 (+4.6%)**		0.4209 (+0.4%)	0.4194
Finnish	MAP	0.3411 (-4.7%)	**0.3796 (+6.1%)**	0.3722 (+4%)	0.369 (+3.1%)	0.3553 (-0.7%)	-	0.3578
	P@5	0.394 (+3.1%)	**0.403 (+5.5%)**	**0.406 (+6.3%)**	**0.4119 (+7.8%)**	0.397 (+3.9%)		0.3821
	P@10	**0.3463 (+11.5%)**	**0.3582 (+15.4%)**	**0.3478 (+12%)**	**0.3448 (+11%)**	**0.3433 (+10.6%)**		0.3105

Table 6: Results showing the performance of MultiPRF with different source and assisting languages using Google Translate for query translation step. The intra-familial affinity could be observed from the elements close to the diagonal.

ease of processing in these languages. (b) *Familial Similarity Between Languages:* We observe that the performance of MultiPRF is good if the assisting language is from the same language family. Birch et al. (2008) show that the language family is a strong predictor of machine translation performance. Hence, the query translation and back translation quality improves if the source and assisting languages belong to the same family. For example, in the Germanic family, the source-assisting language pairs German-English, Dutch-English, Dutch-German and German-Dutch show good performance. Similarly, in Romance family, the performance of French-Spanish confirms this behaviour. In some cases, we observe that MultiPRF scores decent improvements even when the assisting language does not belong to the same language family as witnessed in French-English and English-French. This is primarily due to their strong monolingual MBF performance.

6.3 Effect of Language Family on Back Translation Performance

As already mentioned, the performance of MultiPRF is good if the source and assisting languages belong to the same family. In this section, we verify the above intuition by studying the impact of language family on back translation performance. The experiment designed is as follows: Given a query in source language L_1, the ideal translation in assisting language L_2 is used to compute the query model in L_2 using only the query terms. Then, without performing PRF the query model is directly back translated from L_2 into L_1 and finally documents are re-ranked using this translated feedback model. Since the automatic query translation and PRF steps have been eliminated, the only factor which influences the MultiPRF performance is the back-translation step. This means that the source-assisting language pairs for which the back-translation is good will score a higher performance. The results of the above experiment is shown in Table 8. For each source language, the best performing assisting languages have been highlighted.

Source Lang.	Assisting Language							
	FR	ES	DE	NL	EN	FI	MBF	MPRF
French	-	**0.3686**	0.3113	0.3366	**0.4338**	0.3011	0.4342	0.4535
Spanish	**0.3647**	-	0.3440	0.3476	**0.3954**	0.3036	0.5000	0.4892
German	0.2729	0.2736	-	**0.2951**	0.2107	0.2266	0.4229	0.4576
Dutch	0.2663	0.2836	**0.2902**	-	0.2757	0.2372	0.3968	0.3989

Table 8: Effect of Language Family on Back Translation Performance measured through MultiPRF MAP. 100 Topics from years 2001 and 2002 were used for all languages.

The results show that the performance of closely related languages like French-Spanish and German-Dutch is more when compared to other source-assistant language pairs. This shows that in case of closely related languages, the back-translation step succeeds in adding good terms which are relevant like morphological variants, synonyms and other semantically related terms. Hence, familial closeness of the assisting language helps in boosting the MultiPRF performance. An exception to this trend is English as assisting lan-

Source Lang.		Assisting Language						Source Lang.MBF
		English	German	Dutch	Spanish	French	Finnish	
English	MAP	-	0.4513 (+0.4%)	0.4475 (-0.4%)	**0.4695 (+4.5%)**	0.4665 (+3.8%)	0.4416 (-1.7%)	0.4495
	P@5		0.5104 (+3.0%)	0.5104 (+3.0%)	**0.5343 (+7.8%)**	**0.5403 (+9.0%)**	0.4806 (-3.0%)	0.4955
	P@10		0.4373 (+1.0%)	0.4358 (+0.7%)	**0.4597 (+6.2%)**	**0.4582 (+5.9%)**	0.4164 (-3.8%)	0.4328
German	MAP	**0.4427 (+9.8%)**	-	**0.4306 (+6.8%)**	**0.4404 (+9.2%)**	0.4104 (+1.8%)	0.3993 (-1.0%)	0.4033
	P@5	**0.606 (+18%)**		**0.5672 (+10.5%)**	**0.594 (+15.7%)**	**0.5761 (+12.2%)**	**0.5552 (+8.1%)**	0.5134
	P@10	**0.5373 (+13.2%)**		**0.503 (+6.0%)**	**0.5299 (+11.7%)**	0.494 (+4.1%)	**0.5 (+5.4%)**	0.4746
Dutch	MAP	**0.4361 (+5.0%)**	**0.4344 (+4.6%)**	-	0.4227 (+1.8%)	0.4304 (+3.6%)	0.4134 (-0.5%)	0.4153
	P@5	**0.5761 (+14.2%)**	**0.5552 (+10%)**		**0.5403 (+7.1%)**	**0.5463 (+8.3%)**	**0.5433 (+7.7%)**	0.5045
	P@10	**0.5254 (+12.8%)**	**0.497 (+6.7%)**		0.4776 (+2.6%)	**0.5134 (+10.2%)**	**0.4925 (+5.8%)**	0.4657
Spanish	MAP	0.4665 (-2.9%)	0.4773 (-0.7%)	0.4733 (-1.5%)	-	0.4839 (+0.7%)	0.4412 (-8.2%)	0.4805
	P@5	0.6507 (+1.8%)	0.6448 (+0.9%)	0.6507 (+1.8%)		0.6478 (+1.4%)	0.597 (-6.5%)	0.6388
	P@10	0.5791 (+1.0%)	0.5791 (+1.0%)	0.5761 (+0.5%)		0.5866 (+2.4%)	0.5567 (-2.9%)	0.5731
French	MAP	**0.4591 (+5.4%)**	0.4514 (+3.6%)	0.4409 (+1.2%)	**0.4712 (+8.2%)**	-	0.4354 (0%)	0.4356
	P@5	0.4925 (+3.1%)	0.4776 (0%)	0.4776 (0%)	**0.4995 (+4.6%)**		0.4955 (+3.8%)	0.4776
	P@10	**0.4463 (+6.4%)**	0.4313 (+2.8%)	0.4373 (+4.3%)	**0.4448 (+6.1%)**		0.4209 (+0.3%)	0.4194
Finnish	MAP	0.3733 (+4.3%)	0.3559 (-0.5%)	0.3676 (+2.7%)	0.3594 (+0.4%)	0.371 (+3.7%)	-	0.3578
	P@5	**0.4149 (+8.6%)**	0.385 (+0.7%)	0.388 (+1.6%)	0.388 (+1.6%)	0.3911 (+2.4%)		0.3821
	P@10	**0.3567 (+14.9%)**	0.31 (-0.2%)	**0.3253 (+4.8%)**	0.32 (+3.1%)	0.3239 (+4.3%)		0.3105

Table 7: Results showing the performance of MultiPRF without using automatic query translation *i.e.* by using corresponding original queries in assisting collection. The results show the potential of MultiPRF by establishing a performance upper bound.

guage which shows good performance across both families.

6.4 Multiple Assisting Languages

So far, we have only considered a single assisting language. However, a natural extension to the method which comes to mind, is using multiple assisting languages. In other words, combining the evidence from all the feedback models of more than one assisting language, to get a feedback model which is better than that obtained using a single assisting language. To check how this simple extension works, we performed experiments using a pair of assisting languages. In these experiments for a given source language (from amongst the 6 previously mentioned languages) we tried using all pairs of assisting languages (for each source language, we have 10 pairs possible). To obtain the final model, we simply interpolate all the feedback models with the initial query model, in a similar manner as done in MultiPRF. The results for these experiments are given in Table 9. As we see, out of the 60 possible combinations of source language and assisting language pairs, we obtain improvements of greater than 3% in 16 cases. Here the improvements are with respect to the best model amongst the two MultiPRF models corresponding to each of the two assisting languages, with the same source language. Thus we observe that a simple linear interpolation of models is not the best way of combining evidence from multiple assisting languages. We also observe than when German or Spanish are used as one of the two assisting languages, they are most likely to lead to improvements. A more detailed study of this observation needs to be done to explain this.

Source Language	Assisting Language Pairs with Improvement >3%
English	FR-DE (4.5%), FR-ES (4.8%), DE-NL (+3.1%)
French	EN-DE (4.1%), DE-ES (3.4%), NL-FI (4.8%)
German	None
Spanish	None
Dutch	EN-DE (3.9%), DE-FR (4.1%), FR-ES (3.8%), DE-ES (3.9%)
Finnish	EN-ES (3.2%), FR-DE (4.6%), FR-ES (6.4%), DE-ES (11.2%), DE-NL (4.4%), ES-NL (5.9%)
Total - 16	EN – 3 Pairs; FR – 6 Pairs; DE – 10 Pairs; ES - 8 Pairs; NL – 4 Pairs; FI – 1 Pair

Table 9: Summary of MultiPRF Results with Two Assisting Languages. The improvements described above are with respect to maximum MultiPRF MAP obtained using either L_1 or L_2 alone as assisting language.

7 Conclusion and Future Work

We studied the effect of different *source-assistant* pairs and multiple assisting languages on the performance of MultiPRF. Experiments across a wide range of language pairs with varied degree of familial relationships show that MultiPRF improves performance in most cases with the performance improvement being more pronounced when the source and assisting languages are *closely related*. We also notice that the results are mixed when two assisting languages are used simultaneously. As part of future work, we plan to vary the model interpolation parameters dynamically to improve the performance in case of multiple assisting languages.

Acknowledgements

The first author was supported by a fellowship award from Infosys Technologies Ltd., India. We would like to thank Mr. Vishal Vachhani for his help in running the experiments.

References

Giambattista Amati, Claudio Carpineto, and Giovanni Romano. 2004. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *ECIR '04*, pages 127–137.

Alexandra Birch, Miles Osborne and Philipp Koehn. 2008. Predicting Success in Machine Translation. In *EMNLP '08*, pages 745-754, ACL.

Martin Braschler and Carol Peters. 2004. Cross-Language Evaluation Forum: Objectives, Results, Achievements. *Inf. Retr.*, 7(1-2):7–31.

Martin Braschler and Peter Schäuble. 1998. Multilingual Information Retrieval based on Document Alignment Techniques. In *ECDL '98*, pages 183–197, Springer-Verlag.

Chris Buckley, Gerald Salton, James Allan, and Amit Singhal. 1994. Automatic Query Expansion using SMART : TREC 3. In *TREC-3*, pages 69–80.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting Good Expansion Terms for Pseudo-Relevance Feedback. In *SIGIR '08*, pages 243–250. ACM.

Manoj K. Chinnakotla, Karthik Raman, and Pushpak Bhattacharyya. 2010. Multilingual PRF: English Lends a Helping Hand. In *SIGIR '10*, ACM.

Kevyn Collins-Thompson and Jamie Callan. 2005. Query Expansion Using Random Walk Models. In *CIKM '05*, pages 704–711. ACM.

Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2004. A Framework for Selective Query Expansion. In *CIKM '04*, pages 236–237. ACM.

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two Languages Are More Informative Than One. In *ACL '91*, pages 130–137. ACL.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

T. Susan Dumais, A. Todd Letsche, L. Michael Littman, and K. Thomas Landauer. 1997. Automatic Cross-Language Retrieval Using Latent Semantic Indexing. In *AAAI '97*, pages 18–24.

Wei Gao, John Blitzer, and Ming Zhou. 2008. Using English Information in Non-English Web Search. In *iNEWS '08*, pages 17–24. ACM.

David Hawking, Paul Thistlewaite, and Donna Harman. 1999. Scaling Up the TREC Collection. *Inf. Retr.*, 1(1-2):115–137.

Hieu Hoang, Alexandra Birch, Chris Callison-burch, Richard Zens, Rwth Aachen, Alexandra Constantin, Marcello Federico, Nicola Bertoldi, Chris Dyer, Brooke Cowan, Wade Shen, Christine Moran, and Ondej Bojar. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL '07*, pages 177–180.

P. Jourlin, S. E. Johnson, K. Spärck Jones and P. C. Woodland. 1999. Improving Retrieval on Imperfect Speech Transcriptions (Poster Abstract). In *SIGIR '99*, pages 283–284. ACM.

John Lafferty and Chengxiang Zhai. 2003. Probabilistic Relevance Models Based on Document and Query Generation. *Language Modeling for Information Retrieval*, pages 1–10. Kluwer International Series on IR.

K. Sparck Jones, S. Walker, and S. E. Robertson. 2000. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments. *Inf. Process. Manage.*, 36(6):779–808.

John Lafferty and Chengxiang Zhai. 2001. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *SIGIR '01*, pages 111–119. ACM.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *SIGIR '01*, pages 120–127. ACM.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-Lingual Relevance Models. In *SIGIR '02*, pages 175–182, ACM.

Edgar Meij, Dolf Trieschnigg, Maarten Rijke de, and Wessel Kraaij. 2009. Conceptual Language Models for Domain-specific Retrieval. *Information Processing & Management*, 2009.

Donald Metzler and W. Bruce Croft. 2007. Latent Concept Expansion Using Markov Random Fields. In *SIGIR '07*, pages 311–318. ACM.

Mandar Mitra, Amit Singhal, and Chris Buckley. 1998. Improving Automatic Query Expansion. In *SIGIR '98*, pages 206–214. ACM.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

I. Ounis, G. Amati, Plachouras V., B. He, C. Macdonald, and Johnson. 2005. Terrier Information Retrieval Platform. In *ECIR '05*, volume 3408 of *Lecture Notes in Computer Science*, pages 517–519. Springer.

Koehn Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit '05*.

Stephen Robertson. 2006. On GMAP: and Other Transformations. In *CIKM '06*, pages 78–83. ACM.

Tetsuya Sakai, Toshihiko Manabe, and Makoto Koyama. 2005. Flexible Pseudo-Relevance Feedback Via Selective Sampling. *ACM TALIP*, 4(2):111–135.

Tao Tao and ChengXiang Zhai. 2006. Regularized Estimation of Mixture Models for Robust Pseudo-Relevance Feedback. In *SIGIR '06*, pages 162–169. ACM.

Tuomas Talvensaari, Jorma Laurikkala, Kalervo Järvelin, Martti Juhola, and Heikki Keskustalo. 2007. Creating and Exploiting a Comparable Corpus in Cross-language Information Retrieval. *ACM Trans. Inf. Syst.*, 25(1):4, 2007.

Jrg Tiedemann. 2001. The Use of Parallel Corpora in Monolingual Lexicography - How word alignment can identify morphological and semantic relations. In *COMPLEX '01*, pages 143–151.

Ellen M. Voorhees. 1994. Query Expansion Using Lexical-Semantic Relations. In *SIGIR '94*, pages 61–69. Springer-Verlag.

Ellen Voorhees. 2006. Overview of the TREC 2005 Robust Retrieval Track. In *TREC 2005*, Gaithersburg, MD. NIST.

Dan Wu, Daqing He, Heng Ji, and Ralph Grishman. 2008. A Study of Using an Out-of-Box Commercial MT System for Query Translation in CLIR. In *iNEWS '08*, pages 71–76. ACM.

Jinxi Xu and W. Bruce Croft. 2000. Improving the Effectiveness of Information Retrieval with Local Context Analysis. *ACM Trans. Inf. Syst.*, 18(1):79–112.

Jinxi Xu, Alexander Fraser, and Ralph Weischedel. 2002. Empirical Studies in Strategies for Arabic Retrieval. In *SIGIR '02*, pages 269–274. ACM.

Yang Xu, Gareth J.F. Jones, and Bin Wang. 2009. Query Dependent Pseudo-Relevance Feedback Based on Wikipedia. In *SIGIR '09*, pages 59–66. ACM.

Chengxiang Zhai and John Lafferty. 2001. Model-based Feedback in the Language Modeling approach to Information Retrieval. In *CIKM '01*, pages 403–410. ACM.

Chengxiang Zhai and John Lafferty. 2004. A Study of Smoothing Methods for Language Models applied to Information Retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Wikipedia as Sense Inventory to Improve Diversity in Web Search Results

Celina Santamaría, Julio Gonzalo and **Javier Artiles**
nlp.uned.es
UNED, c/Juan del Rosal, 16, 28040 Madrid, Spain
celina.santamaria@gmail.com julio@lsi.uned.es javart@bec.uned.es

Abstract

Is it possible to use sense inventories to improve Web search results diversity for one word queries? To answer this question, we focus on two broad-coverage lexical resources of a different nature: WordNet, as a de-facto standard used in Word Sense Disambiguation experiments; and Wikipedia, as a large coverage, updated encyclopaedic resource which may have a better coverage of relevant senses in Web pages.

Our results indicate that (i) Wikipedia has a much better coverage of search results, (ii) the distribution of senses in search results can be estimated using the internal graph structure of the Wikipedia and the relative number of visits received by each sense in Wikipedia, and (iii) associating Web pages to Wikipedia senses with simple and efficient algorithms, we can produce modified rankings that cover 70% more Wikipedia senses than the original search engine rankings.

1 Motivation

The application of Word Sense Disambiguation (WSD) to Information Retrieval (IR) has been subject of a significant research effort in the recent past. The essential idea is that, by indexing and matching word senses (or even meanings) , the retrieval process could better handle polysemy and synonymy problems (Sanderson, 2000). In practice, however, there are two main difficulties: (i) for long queries, IR models implicitly perform disambiguation, and thus there is little room for improvement. This is the case with most standard IR benchmarks, such as TREC (trec.nist.gov) or CLEF (www.clef-campaign.org) ad-hoc collections; (ii) for very short queries, disambiguation may not be possible or even desirable. This is often the case with one word and even two word queries in Web search engines.

In Web search, there are at least three ways of coping with ambiguity:

- Promoting diversity in the search results (Clarke et al., 2008): given the query "*oasis*", the search engine may try to include representatives for different senses of the word (such as the Oasis band, the Organization for the Advancement of Structured Information Standards, the online fashion store, etc.) among the top results. Search engines are supposed to handle diversity as one of the multiple factors that influence the ranking.

- Presenting the results as a set of (labelled) clusters rather than as a ranked list (Carpineto et al., 2009).

- Complementing search results with search suggestions (e.g. "*oasis band*", "*oasis fashion store*") that serve to refine the query in the intended way (Anick, 2003).

All of them rely on the ability of the search engine to cluster search results, detecting topic similarities. In all of them, disambiguation is implicit, a side effect of the process but not its explicit target. Clustering may detect that documents about the Oasis band and the Oasis fashion store deal with unrelated topics, but it may as well detect a group of documents discussing why one of the Oasis band members is leaving the band, and another group of documents about Oasis band lyrics; both are different aspects of the broad topic Oasis band. A perfect hierarchical clustering should distinguish between the different Oasis senses at a first level, and then discover different topics within each of the senses.

Is it possible to use sense inventories to improve search results for one word queries? To answer

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1357–1366,
Uppsala, Sweden, 11-16 July 2010.

this question, we will focus on two broad-coverage lexical resources of a different nature: WordNet (Miller et al., 1990), as a de-facto standard used in Word Sense Disambiguation experiments and many other Natural Language Processing research fields; and Wikipedia (www.wikipedia.org), as a large coverage and updated encyclopedic resource which may have a better coverage of relevant senses in Web pages.

Our hypothesis is that, under appropriate conditions, any of the above mechanisms (clustering, search suggestions, diversity) might benefit from an explicit disambiguation (classification of pages in the top search results) using a wide-coverage sense inventory. Our research is focused on four relevant aspects of the problem:

1. Coverage: Are Wikipedia/Wordnet senses representative of search results? Otherwise, trying to make a disambiguation in terms of a fixed sense inventory would be meaningless.

2. If the answer to (1) is positive, the reverse question is also interesting: can we estimate search results diversity using our sense inventories?

3. Sense frequencies: knowing sense frequencies in (search results) Web pages is crucial to have a usable sense inventory. Is it possible to estimate Web sense frequencies from currently available information?

4. Classification: The association of Web pages to word senses must be done with some unsupervised algorithm, because it is not possible to hand-tag training material for every possible query word. Can this classification be done accurately? Can it be effective to promote diversity in search results?

In order to provide an initial answer to these questions, we have built a corpus consisting of 40 nouns and 100 Google search results per noun, manually annotated with the most appropriate Wordnet and Wikipedia senses. Section 2 describes how this corpus has been created, and in Section 3 we discuss WordNet and Wikipedia coverage of search results according to our testbed. As this initial results clearly discard Wordnet as a sense inventory for the task, the rest of the paper mainly focuses on Wikipedia. In Section 4 we estimate search results diversity from our testbed, finding that the use of Wikipedia could substantially improve diversity in the top results. In Section 5 we use the Wikipedia internal link structure and the number of visits per page to estimate relative frequencies for Wikipedia senses, obtaining an estimation which is highly correlated with actual data in our testbed. Finally, in Section 6 we discuss a few strategies to classify Web pages into word senses, and apply the best classifier to enhance diversity in search results. The paper concludes with a discussion of related work (Section 7) and an overall discussion of our results in Section 8.

2 Test Set

2.1 Set of Words

The most crucial step in building our test set is choosing the set of words to be considered. We are looking for words which are susceptible to form a one-word query for a Web search engine, and therefore we should focus on nouns which are used to denote one or more named entities. At the same time we want to have some degree of comparability with previous research on Word Sense Disambiguation, which points to noun sets used in Senseval/SemEval evaluation campaigns[1]. Our budget for corpus annotation was enough for two persons-month, which limited us to handle 40 nouns (usually enough to establish statistically significant differences between WSD algorithms, although obviously limited to reach solid figures about the general behaviour of words in the Web).

With these arguments in mind, we decided to choose: (i) 15 nouns from the Senseval-3 lexical sample dataset, which have been previously employed by (Mihalcea, 2007) in a related experiment (see Section 7); (ii) 25 additional words which satisfy two conditions: they are all ambiguous, and they are all names for music bands in one of their senses (not necessarily the most salient). The Senseval set is: {*argument, arm, atmosphere, bank, degree, difference, disc, image, paper, party, performance, plan, shelter, sort, source*}. The bands set is {*amazon, apple, camel, cell, columbia, cream, foreigner, fox, genesis, jaguar, oasis, pioneer, police, puma, rainbow, shell, skin, sun, tesla, thunder, total, traffic, trapeze, triumph, yes*}.

For each noun, we looked up all its possible senses in WordNet 3.0 and in Wikipedia (using

[1] http://senseval.org

Table 1: Coverage of Search Results: Wikipedia vs. WordNet

	Wikipedia		WordNet	
	# senses available/used	# documents assigned to some sense	# senses available/used	# documents assigned to some sense
Senseval set	242/100	877 (59%)	92/52	696 (46%)
Bands set	640/174	1358 (54%)	78/39	599 (24%)
Total	882/274	2235 (56%)	170/91	1295 (32%)

Wikipedia disambiguation pages). Wikipedia has an average of 22 senses per noun (25.2 in the Bands set and 16.1 in the Senseval set), and Wordnet a much smaller figure, 4.5 (3.12 for the Bands set and 6.13 for the Senseval set). For a conventional dictionary, a higher ambiguity might indicate an excess of granularity; for an encyclopaedic resource such as Wikipedia, however, it is just an indication of larger coverage. Wikipedia entries for *camel* which are not in WordNet, for instance, include the Apache Camel routing and mediation engine, the British rock band, the brand of cigarettes, the river in Cornwall, and the World World War I fighter biplane.

2.2 Set of Documents

We retrieved the 150 first ranked documents for each noun, by submitting the nouns as queries to a Web search engine (Google). Then, for each document, we stored both the snippet (small description of the contents of retrieved document) and the whole HTML document. This collection of documents contain an implicit new inventory of senses, based on Web search, as documents retrieved by a noun query are associated with some sense of the noun. Given that every document in the top Web search results is supposed to be highly relevant for the query word, we assume a "one sense per document" scenario, although we allow annotators to assign more than one sense per document. In general this assumption turned out to be correct except in a few exceptional cases (such as Wikipedia disambiguation pages): only nine documents received more than one WordNet sense, and 44 (1.1% of all annotated pages) received more than one Wikipedia sense.

2.3 Manual Annotation

We implemented an annotation interface which stored all documents and a short description for every Wordnet and Wikipedia sense. The annotators had to decide, for every document, whether there was one or more appropriate senses in each of the dictionaries. They were instructed to provide annotations for 100 documents per name; if an URL in the list was corrupt or not available, it had to be discarded. We provided 150 documents per name to ensure that the figure of 100 usable documents per name could be reached without problems.

Each judge provided annotations for the 4,000 documents in the final data set. In a second round, they met and discussed their independent annotations together, reaching a consensus judgement for every document.

3 Coverage of Web Search Results: Wikipedia vs Wordnet

Table 1 shows how Wikipedia and Wordnet cover the senses in search results. We report each noun subset separately (*Senseval* and *bands* subsets) as well as aggregated figures.

The most relevant fact is that, unsurprisingly, Wikipedia senses cover much more search results (56%) than Wordnet (32%). If we focus on the top ten results, in the bands subset (which should be more representative of plausible web queries) Wikipedia covers 68% of the top ten documents. This is an indication that it can indeed be useful for promoting diversity or help clustering search results: even if 32% of the top ten documents are not covered by Wikipedia, it is still a representative source of senses in the top search results.

We have manually examined all documents in the top ten results that are not covered by Wikipedia: a majority of the missing senses consists of names of (generally not well-known) companies (45%) and products or services (26%); the other frequent type (12%) of non annotated document is disambiguation pages (from Wikipedia and also from other dictionaries).

It is also interesting to examine the degree of overlap between Wikipedia and Wordnet senses. Being two different types of lexical resource, they might have some degree of complementarity. Table 2 shows, however, that this is not the case: most of the (annotated) documents either fit Wikipedia senses (26%) or both Wikipedia and Wordnet (29%), and just 3% fit Wordnet only.

Table 2: Overlap between Wikipedia and Wordnet in Search Results

	# documents annotated with			
	Wikipedia & Wordnet	Wikipedia only	Wordnet only	none
Senseval set	607 (40%)	270 (18%)	89 (6%)	534 (36%)
Bands set	572 (23%)	786 (31%)	27 (1%)	1115 (45%)
Total	1179 (29%)	1056 (26%)	116 (3%)	1649 (41%)

Therefore, Wikipedia seems to extend the coverage of Wordnet rather than providing complementary sense information. If we wanted to extend the coverage of Wikipedia, the best strategy seems to be to consider lists of companies, products and services, rather than complementing Wikipedia with additional sense inventories.

4 Diversity in Google Search Results

Once we know that Wikipedia senses are a representative subset of actual Web senses (covering more than half of the documents retrieved by the search engine), we can test how well search results respect diversity in terms of this subset of senses.

Table 3 displays the number of different senses found at different depths in the search results rank, and the average proportion of total senses that they represent. These results suggest that diversity is not a major priority for ranking results: the top ten results only cover, in average, 3 Wikipedia senses (while the average number of senses listed in Wikipedia is 22). When considering the first 100 documents, this number grows up to 6.85 senses per noun.

Another relevant figure is the frequency of the most frequent sense for each word: in average, 63% of the pages in search results belong to the most frequent sense of the query word. This is roughly comparable with most frequent sense figures in standard annotated corpora such as Semcor (Miller et al., 1993) and the Senseval/Semeval data sets, which suggests that diversity may not play a major role in the current Google ranking algorithm.

Of course this result must be taken with care, because variability between words is high and unpredictable, and we are using only 40 nouns for our experiment. But what we have is a positive indication that Wikipedia could be used to improve diversity or cluster search results: potentially the first top ten results could cover 6.15 different senses in average (see Section 6.5), which would be a substantial growth.

5 Sense Frequency Estimators for Wikipedia

Wikipedia disambiguation pages contain no systematic information about the relative importance of senses for a given word. Such information, however, is crucial in a lexicon, because sense distributions tend to be skewed, and knowing them can help disambiguation algorithms.

We have attempted to use two estimators of expected sense distribution:

- Internal relevance of a word sense, measured as incoming links for the URL of a given sense in Wikipedia.
- External relevance of a word sense, measured as the number of visits for the URL of a given sense (as reported in http://stats.grok.se).

The number of internal incoming links is expected to be relatively stable for Wikipedia articles. As for the number of visits, we performed a comparison of the number of visits received by the bands noun subset in May, June and July 2009, finding a stable-enough scenario with one notorious exception: the number of visits to the noun Tesla raised dramatically in July, because July 10 was the anniversary of the birth of Nicola Tesla, and a special Google logo directed users to the Wikipedia page for the scientist.

We have measured correlation between the relative frequencies derived from these two indicators and the actual relative frequencies in our testbed. Therefore, for each noun w and for each sense w_i, we consider three values: (i) proportion of documents retrieved for w which are manually assigned to each sense w_i; (ii) inlinks(w_i): relative amount of incoming links to each sense w_i; and (iii) visits(w_i): relative number of visits to the URL for each sense w_i.

We have measured the correlation between these three values using a linear regression correlation coefficient, which gives a correlation value of .54 for the number of visits and of .71 for the number of incoming links. Both estimators seem

Table 3: Diversity in Search Results according to Wikipedia

	average # senses in search results			average coverage of Wikipedia senses		
	Bands set	Senseval set	Total	Bands set	Senseval set	Total
First 10 docs	2.88	3.2	3.00	.21	.21	.21
First 25	4.44	4.8	4.58	.28	.33	.30
First 50	5.56	5.47	5.53	.33	.36	.34
First 75	6.56	6.33	6.48	.37	.43	.39
First 100	6.96	6.67	6.85	.38	.45	.41

to be positively correlated with real relative frequencies in our testbed, with a strong preference for the number of links.

We have experimented with weighted combinations of both indicators, using weights of the form $(k, 1-k), k \in \{0, 0.1, 0.2 \ldots 1\}$, reaching a maximal correlation of .73 for the following weights:

$$\text{freq}(w_i) = 0.9 * \text{inlinks}(w_i) + 0.1 * \text{visits}(w_i) \quad (1)$$

This weighted estimator provides a slight advantage over the use of incoming links only (.73 vs .71). Overall, we have an estimator which has a strong correlation with the distribution of senses in our testbed. In the next section we will test its utility for disambiguation purposes.

6 Association of Wikipedia Senses to Web Pages

We want to test whether the information provided by Wikipedia can be used to classify search results accurately. Note that we do not want to consider approaches that involve a manual creation of training material, because they can't be used in practice.

Given a Web page p returned by the search engine for the query w, and the set of senses $w_1 \ldots w_n$ listed in Wikipedia, the task is to assign the best candidate sense to p. We consider two different techniques:

- A basic Information Retrieval approach, where the documents and the Wikipedia pages are represented using a Vector Space Model (VSM) and compared with a standard cosine measure. This is a basic approach which, if successful, can be used efficiently to classify search results.

- An approach based on a state-of-the-art supervised WSD system, extracting training examples automatically from Wikipedia content.

We also compute two baselines:

- A random assignment of senses (precision is computed as the inverse of the number of senses, for every test case).

- A most frequent sense heuristic which uses our estimation of sense frequencies and assigns the same sense (the most frequent) to all documents.

Both are naive baselines, but it must be noted that the most frequent sense heuristic is usually hard to beat for unsupervised WSD algorithms in most standard data sets.

We now describe each of the two main approaches in detail.

6.1 VSM Approach

For each word sense, we represent its Wikipedia page in a (unigram) vector space model, assigning standard tf*idf weights to the words in the document. idf weights are computed in two different ways:

1. Experiment **VSM** computes inverse document frequencies in the collection of retrieved documents (for the word being considered).

2. Experiment **VSM-GT** uses the statistics provided by the Google Terabyte collection (Brants and Franz, 2006), i.e. it replaces the collection of documents with statistics from a representative snapshot of the Web.

3. Experiment **VSM-mixed** combines statistics from the collection and from the Google Terabyte collection, following (Chen et al., 2009).

The document p is represented in the same vector space as the Wikipedia senses, and it is compared with each of the candidate senses w_i via the cosine similarity metric (we have experimented

with other similarity metrics such as χ^2, but differences are irrelevant). The sense with the highest similarity to p is assigned to the document. In case of ties (which are rare), we pick the first sense in the Wikipedia disambiguation page (which in practice is like a random decision, because senses in disambiguation pages do not seem to be ordered according to any clear criteria).

We have also tested a variant of this approach which uses the estimation of sense frequencies presented above: once the similarities are computed, we consider those cases where two or more senses have a similar score (in particular, all senses with a score greater or equal than 80% of the highest score). In that cases, instead of using the small similarity differences to select a sense, we pick up the one which has the largest frequency according to our estimator. We have applied this strategy to the best performing system, VSM-GT, resulting in experiment **VSM-GT+freq**.

6.2 WSD Approach

We have used TiMBL (Daelemans et al., 2001), a state-of-the-art supervised WSD system which uses Memory-Based Learning. The key, in this case, is how to extract learning examples from the Wikipedia automatically. For each word sense, we basically have three sources of examples: (i) occurrences of the word in the Wikipedia page for the word sense; (ii) occurrences of the word in Wikipedia pages pointing to the page for the word sense; (iii) occurrences of the word in external pages linked in the Wikipedia page for the word sense.

After an initial manual inspection, we decided to discard external pages for being too noisy, and we focused on the first two options. We tried three alternatives:

- **TiMBL-core** uses only the examples found in the page for the sense being trained.
- **TiMBL-inlinks** uses the examples found in Wikipedia pages pointing to the sense being trained.
- **TiMBL-all** uses both sources of examples.

In order to classify a page p with respect to the senses for a word w, we first disambiguate all occurrences of w in the page p. Then we choose the sense which appears most frequently in the page according to TiMBL results. In case of ties we pick up the first sense listed in the Wikipedia disambiguation page.

We have also experimented with a variant of the approach that uses our estimation of sense frequencies, similarly to what we did with the VSM approach. In this case, (i) when there is a tie between two or more senses (which is much more likely than in the VSM approach), we pick up the sense with the highest frequency according to our estimator; and (ii) when no sense reaches 30% of the cases in the page to be disambiguated, we also resort to the most frequent sense heuristic (among the candidates for the page). This experiment is called TiMBL-core+freq (we discarded "inlinks" and "all" versions because they were clearly worse than "core").

6.3 Classification Results

Table 4 shows classification results. The accuracy of systems is reported as precision, i.e. the number of pages correctly classified divided by the total number of predictions. This is approximately the same as recall (correctly classified pages divided by total number of pages) for our systems, because the algorithms provide an answer for every page containing text (actual coverage is 94% because some pages only contain text as part of an image file such as photographs and logotypes).

Table 4: Classification Results

Experiment	Precision
random	.19
most frequent sense (estimation)	.46
TiMBL-core	.60
TiMBL-inlinks	.50
TiMBL-all	.58
TiMBL-core+freq	**.67**
VSM	.67
VSM-GT	.68
VSM-mixed	.67
VSM-GT+freq	**.69**

All systems are significantly better than the random and most frequent sense baselines (using $p < 0.05$ for a standard t-test). Overall, both approaches (using TiMBL WSD machinery and using VSM) lead to similar results (.67 vs. .69), which would make VSM preferable because it is a simpler and more efficient approach. Taking a

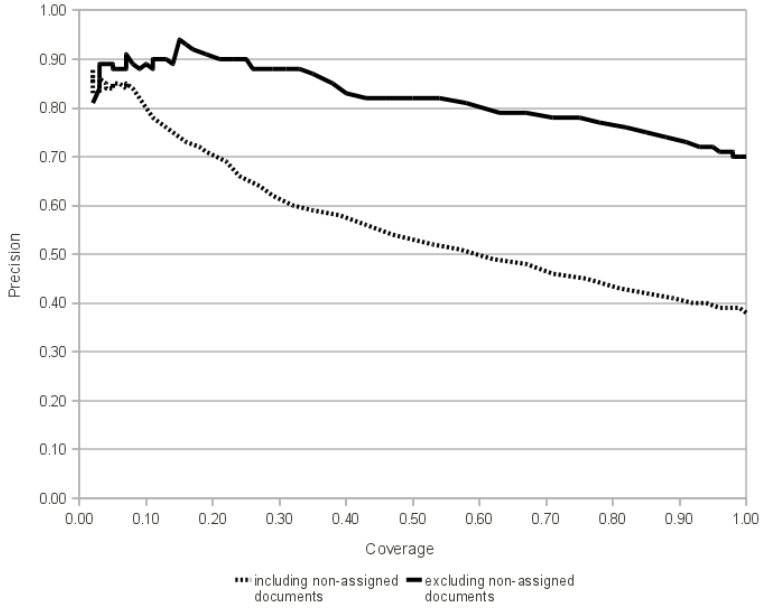


Figure 1: Precision/Coverage curves for VSM-GT+freq classification algorithm

closer look at the results with TiMBL, there are a couple of interesting facts:

- There is a substantial difference between using only examples taken from the Wikipedia Web page for the sense being trained (TiMBL-core, .60) and using examples from the Wikipedia pages pointing to that page (TiMBL-inlinks, .50). Examples taken from related pages (even if the relationship is close as in this case) seem to be too noisy for the task. This result is compatible with findings in (Santamaría et al., 2003) using the Open Directory Project to extract examples automatically.

- Our estimation of sense frequencies turns out to be very helpful for cases where our TiMBL-based algorithm cannot provide an answer: precision rises from .60 (TiMBL-core) to .67 (TiMBL-core+freq). The difference is statistically significant ($p < 0.05$) according to the t-test.

As for the experiments with VSM, the variations tested do not provide substantial improvements to the baseline (which is .67). Using idf frequencies obtained from the Google Terabyte corpus (instead of frequencies obtained from the set of retrieved documents) provides only a small improvement (VSM-GT, .68), and adding the estimation of sense frequencies gives another small improvement (.69). Comparing the baseline VSM with the optimal setting (VSM-GT+freq), the difference is small (.67 vs .69) but relatively robust ($p = 0.066$ according to the t-test).

Remarkably, the use of frequency estimations is very helpful for the WSD approach but not for the SVM one, and they both end up with similar performance figures; this might indicate that using frequency estimations is only helpful up to certain precision ceiling.

6.4 Precision/Coverage Trade-off

All the above experiments are done at maximal coverage, i.e., all systems assign a sense for every document in the test collection (at least for every document with textual content). But it is possible to enhance search results diversity without annotating every document (in fact, not every document can be assigned to a Wikipedia sense, as we have discussed in Section 3). Thus, it is useful to investigate which is the precision/coverage trade-off in our dataset. We have experimented with the best performing system (VSM-GT+freq), introducing a similarity threshold: assignment of a document to a sense is only done if the similarity of the document to the Wikipedia page for the sense exceeds the similarity threshold.

We have computed precision and coverage for every threshold in the range $[0.00 - 0.90]$ (beyond 0.90 coverage was null) and represented the results in Figure 1 (solid line). The graph shows that we

can classify around 20% of the documents with a precision above .90, and around 60% of the documents with a precision of .80.

Note that we are reporting disambiguation results using a conventional WSD test set, i.e., one in which every test case (every document) has been manually assigned to some Wikipedia sense. But in our Web Search scenario, 44% of the documents were not assigned to any Wikipedia sense: in practice, our classification algorithm would have to cope with all this noise as well. Figure 1 (dotted line) shows how the precision/coverage curve is affected when the algorithm attempts to disambiguate all documents retrieved by Google, whether they can in fact be assigned to a Wikipedia sense or not. At a coverage of 20%, precision drops approximately from .90 to .70, and at a coverage of 60% it drops from .80 to .50. We now address the question of whether this performance is good enough to improve search results diversity in practice.

6.5 Using Classification to Promote Diversity

We now want to estimate how the reported classification accuracy may perform in practice to enhance diversity in search results. In order to provide an initial answer to this question, we have re-ranked the documents for the 40 nouns in our testbed, using our best classifier (VSM-GT+freq) and making a list of the top-ten documents with the primary criterion of maximising the number of senses represented in the set, and the secondary criterion of maximising the similarity scores of the documents to their assigned senses. The algorithm proceeds as follows: we fill each position in the rank (starting at rank 1), with the document which has the highest similarity to some of the senses which are not yet represented in the rank; once all senses are represented, we start choosing a second representative for each sense, following the same criterion. The process goes on until the first ten documents are selected.

We have also produced a number of alternative rankings for comparison purposes:

- **clustering (centroids)**: this method applies Hierarchical Agglomerative Clustering – which proved to be the most competitive clustering algorithm in a similar task (Artiles et al., 2009) – to the set of search results, forcing the algorithm to create ten clusters. The centroid of each cluster is then selected as one of the top ten documents in the new rank.

- **clustering (top ranked)**: Applies the same clustering algorithm, but this time the top ranked document (in the original Google rank) of each cluster is selected.

- **random**: Randomly selects ten documents from the set of retrieved results.

- **upper bound**: This is the maximal diversity that can be obtained in our testbed. Note that coverage is not 100%, because some words have more than ten meanings in Wikipedia and we are only considering the top ten documents.

Table 5: Enhancement of Search Results Diversity

rank@10	# senses	coverage
Original rank	2.80	49%
Wikipedia	**4.75**	**77%**
clustering (centroids)	2.50	42%
clustering (top ranked)	2.80	46%
random	2.45	43%
upper bound	6.15	97%

All experiments have been applied on the full set of documents in the testbed, including those which could not be annotated with any Wikipedia sense. Coverage is computed as the ratio of senses that appear in the top ten results compared to the number of senses that appear in all search results.

Results are presented in Table 5. Note that diversity in the top ten documents increases from an average of 2.80 Wikipedia senses represented in the original search engine rank, to 4.75 in the modified rank (being 6.15 the upper bound), with the coverage of senses going from 49% to 77%. With a simple VSM algorithm, the coverage of Wikipedia senses in the top ten results is 70% larger than in the original ranking.

Using Wikipedia to enhance diversity seems to work much better than clustering: both strategies to select a representative from each cluster are unable to improve the diversity of the original ranking. Note, however, that our evaluation has a bias towards using Wikipedia, because only Wikipedia senses are considered to estimate diversity.

Of course our results do not imply that the Wikipedia modified rank is better than the original

Google rank: there are many other factors that influence the final ranking provided by a search engine. What our results indicate is that, with simple and efficient algorithms, Wikipedia can be used as a reference to improve search results diversity for one-word queries.

7 Related Work

Web search results clustering and diversity in search results are topics that receive an increasing attention from the research community. Diversity is used both to represent sub-themes in a broad topic, or to consider alternative interpretations for ambiguous queries (Agrawal et al., 2009), which is our interest here. Standard IR test collections do not usually consider ambiguous queries, and are thus inappropriate to test systems that promote diversity (Sanderson, 2008); it is only recently that appropriate test collections are being built, such as (Paramita et al., 2009) for image search and (Artiles et al., 2009) for person name search. We see our testbed as complementary to these ones, and expect that it can contribute to foster research on search results diversity.

To our knowledge, Wikipedia has not explicitly been used before to promote diversity in search results; but in (Gollapudi and Sharma, 2009), it is used as a gold standard to evaluate diversification algorithms: given a query with a Wikipedia disambiguation page, an algorithm is evaluated as promoting diversity when different documents in the search results are semantically similar to different Wikipedia pages (describing the alternative senses of the query). Although semantic similarity is measured automatically in this work, our results confirm that this evaluation strategy is sound, because Wikipedia senses are indeed representative of search results.

(Clough et al., 2009) analyses query diversity in a Microsoft Live Search, using click entropy and query reformulation as diversity indicators. It was found that at least 9.5% - 16.2% of queries could benefit from diversification, although no correlation was found between the number of senses of a word in Wikipedia and the indicators used to discover diverse queries. This result does not discard, however, that queries where applying diversity is useful cannot benefit from Wikipedia as a sense inventory.

In the context of clustering, (Carmel et al., 2009) successfully employ Wikipedia to enhance automatic cluster labeling, finding that Wikipedia labels agree with manual labels associated by humans to a cluster, much more than with significant terms that are extracted directly from the text. In a similar line, both (Gabrilovich and Markovitch, 2007) and (Syed et al., 2008) provide evidence suggesting that categories of Wikipedia articles can successfully describe common concepts in documents.

In the field of Natural Language Processing, there has been successful attempts to connect Wikipedia entries to Wordnet senses: (Ruiz-Casado et al., 2005) reports an algorithm that provides an accuracy of 84%. (Mihalcea, 2007) uses internal Wikipedia hyperlinks to derive sense-tagged examples. But instead of using Wikipedia directly as sense inventory, Mihalcea then manually maps Wikipedia senses into Wordnet senses (claiming that, at the time of writing the paper, Wikipedia did not consistently report ambiguity in disambiguation pages) and shows that a WSD system based on acquired sense-tagged examples reaches an accuracy well beyond an (informed) most frequent sense heuristic.

8 Conclusions

We have investigated whether generic lexical resources can be used to promote diversity in Web search results for one-word, ambiguous queries. We have compared WordNet and Wikipedia and arrived to a number of conclusions: (i) unsurprisingly, Wikipedia has a much better coverage of senses in search results, and is therefore more appropriate for the task; (ii) the distribution of senses in search results can be estimated using the internal graph structure of the Wikipedia and the relative number of visits received by each sense in Wikipedia, and (iii) associating Web pages to Wikipedia senses with simple and efficient algorithms, we can produce modified rankings that cover 70% more Wikipedia senses than the original search engine rankings.

We expect that the testbed created for this research will complement the - currently short - set of benchmarking test sets to explore search results diversity and query ambiguity. Our testbed is publicly available for research purposes at http://nlp.uned.es.

Our results endorse further investigation on the use of Wikipedia to organize search results. Some limitations of our research, however, must be

noted: (i) the nature of our testbed (with every search result manually annotated in terms of two sense inventories) makes it too small to extract solid conclusions on Web searches (ii) our work does not involve any study of diversity from the point of view of Web users (i.e. when a Web query addresses many different use needs in practice); research in (Clough et al., 2009) suggests that word ambiguity in Wikipedia might not be related with diversity of search needs; (iii) we have tested our classifiers with a simple re-ordering of search results to test how much diversity can be improved, but a search results ranking depends on many other factors, some of them more crucial than diversity; it remains to be tested how can we use document/Wikipedia associations to improve search results clustering (for instance, providing seeds for the clustering process) and to provide search suggestions.

Acknowledgments

This work has been partially funded by the Spanish Government (project INES/Text-Mess) and the Xunta de Galicia.

References

R. Agrawal, S. Gollapudi, A. Halverson, and S. Leong. 2009. Diversifying Search Results. In *Proc. of WSDM'09*. ACM.

P. Anick. 2003. Using Terminological Feedback for Web Search Refinement : a Log-based Study. In *Proc. ACM SIGIR 2003*, pages 88–95. ACM New York, NY, USA.

J. Artiles, J. Gonzalo, and S. Sekine. 2009. WePS 2 Evaluation Campaign: overview of the Web People Search Clustering Task. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference. 2009.*

T. Brants and A. Franz. 2006. Web 1T 5-gram, version 1. Philadelphia: Linguistic Data Consortium.

D. Carmel, H. Roitman, and N. Zwerdling. 2009. Enhancing Cluster Labeling using Wikipedia. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146. ACM.

C. Carpineto, S. Osinski, G. Romano, and Dawid Weiss. 2009. A Survey of Web Clustering Engines. *ACM Computing Surveys*, 41(3).

Y. Chen, S. Yat Mei Lee, and C. Huang. 2009. PolyUHK: A Robust Information Extraction System for Web Personal Names. In *Proc. WWW'09 (WePS-2 Workshop)*. ACM.

C. Clarke, M. Kolla, G. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. 2008. Novelty and Diversity in Information Retrieval Evaluation. In *Proc. SIGIR'08*, pages 659–666. ACM.

P. Clough, M. Sanderson, M. Abouammoh, S. Navarro, and M. Paramita. 2009. Multiple Approaches to Analysing Query Diversity. In *Proc. of SIGIR 2009*. ACM.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2001. TiMBL: Tilburg Memory Based Learner, version 4.0, Reference Guide. Technical report, University of Antwerp.

E. Gabrilovich and S. Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India*.

S. Gollapudi and A. Sharma. 2009. An Axiomatic Approach for Result Diversification. In *Proc. WWW 2009*, pages 381–390. ACM New York, NY, USA.

R. Mihalcea. 2007. Using Wikipedia for Automatic Word Sense Disambiguation. In *Proceedings of NAACL HLT*, volume 2007.

G. Miller, C. R. Beckwith, D. Fellbaum, Gross, and K. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicograph*, 3(4).

G.A Miller, C. Leacock, R. Tengi, and Bunker R. T. 1993. A Semantic Concordance. In *Proceedings of the ARPA WorkShop on Human Language Technology. San Francisco, Morgan Kaufman.*

M. Paramita, M. Sanderson, and P. Clough. 2009. Diversity in Photo Retrieval: Overview of the ImageCLEFPhoto task 2009. *CLEF working notes*, 2009.

M. Ruiz-Casado, E. Alfonseca, and P. Castells. 2005. Automatic Assignment of Wikipedia Encyclopaedic Entries to Wordnet Synsets. *Advances in Web Intelligence*, 3528:380–386.

M. Sanderson. 2000. Retrieving with Good Sense. *Information Retrieval*, 2(1):49–69.

M. Sanderson. 2008. Ambiguous Queries: Test Collections Need More Sense. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 499–506. ACM New York, NY, USA.

C. Santamaría, J. Gonzalo, and F. Verdejo. 2003. Automatic Association of Web Directories to Word Senses. *Computational Linguistics*, 29(3):485–502.

Z. S. Syed, T. Finin, and Joshi. A. 2008. Wikipedia as an Ontology for Describing Documents. In *Proc. ICWSM'08*.

A Unified Graph Model for Sentence-based Opinion Retrieval

Binyang Li, Lanjun Zhou, Shi Feng, Kam-Fai Wong
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
{byli, ljzhou, sfeng, kfwong}@se.cuhk.edu.hk

Abstract

There is a growing research interest in opinion retrieval as on-line users' opinions are becoming more and more popular in business, social networks, etc. Practically speaking, the goal of opinion retrieval is to retrieve documents, which entail opinions or comments, relevant to a target subject specified by the user's query. A fundamental challenge in opinion retrieval is information representation. Existing research focuses on document-based approaches and documents are represented by *bag-of-word*. However, due to loss of contextual information, this representation fails to capture the associative information between an opinion and its corresponding target. It cannot distinguish different degrees of a sentiment word when associated with different targets. This in turn seriously affects opinion retrieval performance. In this paper, we propose a sentence-based approach based on a new information representation, namely topic-sentiment word pair, to capture intra-sentence contextual information between an opinion and its target. Additionally, we consider inter-sentence information to capture the relationships among the opinions on the same topic. Finally, the two types of information are combined in a unified graph-based model, which can effectively rank the documents. Compared with existing approaches, experimental results on the COAE08 dataset showed that our graph-based model achieved significant improvement.

1 Introduction

In recent years, there is a growing interest in sharing personal opinions on the Web, such as product reviews, economic analysis, political polls, etc. These opinions cannot only help independent users make decisions, but also obtain valuable feedbacks (Pang et al., 2008). Opinion oriented research, including sentiment classification, opinion extraction, opinion question answering, and opinion summarization, etc. are receiving growing attention (Wilson, et al., 2005; Liu et al., 2005; Oard et al., 2006). However, most existing works concentrate on analyzing opinions expressed in the documents, and none on how to represent the information needs required to retrieve opinionated documents. In this paper, we focus on opinion retrieval, whose goal is to find a set of documents containing not only the query keyword(s) but also the relevant opinions. This requirement brings about the challenge on how to represent information needs for effective opinion retrieval.

In order to solve the above problem, previous work adopts a 2-stage approach. In the first stage, relevant documents are determined and ranked by a score, i.e. *tf-idf* value. In the second stage, an opinion score is generated for each relevant document (Macdonald and Ounis, 2007; Oard et al., 2006). The opinion score can be acquired by either machine learning-based sentiment classifiers, such as SVM (Zhang and Yu, 2007), or a sentiment lexicons with weighted scores from training documents (Amati et al., 2007; Hannah et al., 2007; Na et al., 2009). Finally, an overall score combining the two is computed by using a score function, e.g. linear combination, to re-rank the retrieved documents.

Retrieval in the 2-stage approach is based on document and document is represented by *bag-of-word*. This representation, however, can only ensure that there is at least one opinion in each relevant document, but it cannot determine the relevance pairing of individual opinion to its target. In general, by simply representing a document in *bag-of-word*, contextual information i.e. the corresponding target of an opinion, is neglected. This may result in possible mismatch between an opinion and a target and in turn affects opinion retrieval performance. By the same token, the effect to documents consisting of mul-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1367–1375,
Uppsala, Sweden, 11-16 July 2010.

tiple topics, which is common in blogs and on-line reviews, is also significant. In this setting, even if a document is regarded opinionated, it cannot ensure that all opinions in the document are indeed relevant to the target concerned. Therefore, we argue that existing information representation i.e. *bag-of-word*, cannot satisfy the information needs for opinion retrieval.

In this paper, we propose to handle opinion retrieval in the granularity of sentence. It is observed that a complete opinion is always expressed in one sentence, and the relevant target of the opinion is mostly the one found in it. Therefore, it is crucial to maintain the associative information between an opinion and its target within a sentence. We define the notion of a topic-sentiment word pair, which is composed of a topic term (i.e. the target) and a sentiment word (i.e. opinion) of a sentence. Word pairs can maintain intra-sentence contextual information to express the potential relevant opinions. In addition, inter-sentence contextual information is also captured by word pairs to represent the relationship among opinions on the same topic. In practice, the inter-sentence information reflects the degree of a word pair. Finally, we combine both intra-sentence and inter-sentence contextual information to construct a unified undirected graph to achieve effective opinion retrieval.

The rest of the paper is organized as follows. In Section 2, we describe the motivation of our approach. Section 3 presents a novel unified graph-based model for opinion retrieval. We evaluated our model and the results are presented in Section 4. We review related works on opinion retrieval in Section 5. Finally, in Section 6, the paper is concluded and future work is suggested.

2 Motivation

In this section, we start from briefly describing the objective of opinion retrieval. We then illustrate the limitations of current opinion retrieval approaches, and analyze the motivation of our method.

2.1 Formal Description of Problem

Opinion retrieval was first presented in the TREC 2006 Blog track, and the objective is to retrieve documents that express an opinion about a given target. The opinion target can be a "traditional" named entity (e.g. a name of person, location, or organization, etc.), a concept (e.g. a type of technology), or an event (e.g. presidential election). The topic of the document is not required to be the same as the target, but an opinion about the target has to be presented in the document or one of the comments to the document (Macdonald and Ounis, 2006). Therefore, in this paper we regard the information needs for opinion retrieval as *relevant opinion*.

2.2 Motivation of Our Approach

In traditional information retrieval (IR) *bag-of-word* representation is the most common way to express information needs. However, in opinion retrieval, information need target at *relevant opinion*, and this renders *bag-of-word* representation ineffective.

Consider the example in Figure 1. There are three sentences *A*, *B*, and *C* in a document d_i. Now given an opinion-oriented query Q related to '*Avatar'*. According to the conventional 2-stage opinion retrieval approach, d_i is represented by a *bag-of-word*. Among the words, there is a topic term *Avatar* (t_1) occurring twice, i.e. *Avatar in A* and *Avatar in C*, and two sentiment words *comfortable* (o_1) and *favorite* (o_2) (refer to Figure 2 (a)). In order to rank this document, an overall score of the document d_i is computed by a simple combination of the relevant score ($Score_{\text{rel}}$) and the opinion score ($Score_{\text{op}}$), e.g. equal weighted linear combination, as follows.

$$Score_{\text{doc}} = Score_{\text{rel}} + \quad Score_{\text{op}}$$

For simplicity, we let $Score_{\text{rel}} = tf_{\text{Q}} \times idf_{\text{Q}}$, and $Score_{\text{op}}$ be computed by using lexicon-based method: $Score_{\text{op}} = weight_{\text{comfortable}} + weight_{\text{favorite}}$.

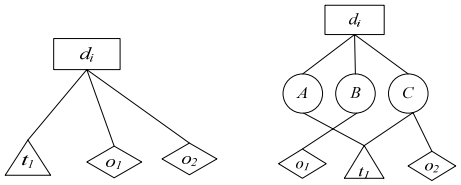
> A. 阿凡达明日将在中国上映。
> Tomorrow, Avatar will be shown in China.
> B. 我预订到了IMAX影院中最舒服的位子。
> I've reserved a comfortable seat in IMAX.
> C. 阿凡达是我最喜欢的一部 3D 电影。
> Avatar is my favorite 3D movie.

Figure 1: A retrieved document d_i on the target '*Avatar'*.

Although *bag-of-word* representation achieves good performance in retrieving relevant documents, our study shows that it cannot satisfy the information needs for retrieval of *relevant opinion*. It suffers from the following limitations:

(1) It cannot maintain contextual information; thus, an opinion may not be related to the target of the retrieved document is neglected. In this example, only the opinion *favorite* (o_2) on *Avatar in C* is the *relevant opinion*. But due to loss of contextual information between the opinion and its corresponding target, *Avatar in A* and *com-*

fortable (o_1) are also regarded as *relevant opinion* mistakenly, creating a false positive. In reality *comfortable* (o_1) describes "the seats in IMAX", which is an irrelevant opinion, and sentence *A* is a factual statement rather than an opinion statement.



(a) (b)

Figure 2: Two kinds of information representation of opinion retrieval. (t_1='*Avatar*' o_1= '*comfortable*', o_2='*favorite*')

(1) Current approaches cannot capture the relationship among opinions about the same topic. Suppose there is another document including sentence *C* which expresses the same opinion on *Avatar*. Existing information representation simply does not cater for the two identical opinions from different documents. In addition, if many documents contain opinions on *Avatar*, the relationship among them is not clearly represented by existing approaches.

In this paper, we process opinion retrieval in the granularity of sentence as we observe that a complete opinion always exists within a sentence (refer to Figure 2 (b)). To represent a *relevant opinion*, we define the notion of topic-sentiment word pair, which consists of a topic term and a sentiment word. A word pair maintains the associative information between the two words, and enables systems to draw up the relationship among all the sentences with the same opinion on an identical target. This relationship information can identify all documents with sentences including the sentiment words and to determine the contributions of such words to the target (topic term). Furthermore, based on word pairs, we designed a unified graph-based method for opinion retrieval (see later in Section 3).

3 Graph-based model

3.1 Basic Idea

Different from existing approaches which simply make use of document relevance to reflect the relevance of opinions embedded in them, our approach concerns more on identifying the relevance of individual opinions. Intuitively, we believed that the more *relevant opinions* appear in a document, the more relevant is that document for subsequent opinion analysis operations. Further, since the lexical scope of an opinion does not usually go beyond a sentence, we propose to handle opinion retrieval in the granularity of sentence.

Without loss of generality, we assume that there is a document set $\mathcal{D} = \{d_1, d_2, d_3, \cdots, d_n\}$, and a specific query $\mathcal{Q} = \{q_1, q_2, q_3, \cdots, q_z\}$, where $q_1, q_2, q_3, \cdots, q_z$ are query keywords. Opinion retrieval aims at retrieving documents from $\mathcal{D}$ with *relevant opinion* about the query $\mathcal{Q}$. In addition, we construct a sentiment word lexicon V_o and a topic term lexicon V_t (see Section 4). To maintain the associative information between the target and the opinion, we consider the document set as a *bag of sentences*, and define a sentence set as $\mathcal{S} = \{s_1, s_2, s_3, \cdots, s_N\}$. For each sentence, we capture the intra-sentence information through the topic-sentiment word pair.

Definition 1. topic-sentiment word pair p_{ij} consists of two elements, one is from V_t, and the other one is from V_o.

$$p_{ij} = \{< t_i, o_j > | t_i \in V_t, o_j \in V_o)\}.$$

The topic term from V_t determines relevance by the query term matching, and the sentiment word from V_o is used to express an opinion. We use the word pair to maintain the associative information between the topic term and the opinion word (also referred to as sentiment word). The word pair is used to identify a *relevant opinion* in a sentence. In Figure 2 (b), t_1, i.e. *Avatar in C*, is a topic term relevant to the query, and o_2 ('*favorite*') is supposed to be an opinion; and the word pair < t_1, o_2> indicates sentence *C* contains a *relevant opinion*. Similarly, we map each sentence in word pairs by the following rule, and express the intra-sentence information using word pairs.

For each sentiment word of a sentence, we choose the topic term with minimum distance as the other element of the word pair:

$$s_l \rightarrow \{< t_i, o_j > | t_i = \min Dist(t_i, o_j) \text{ for each } o_j\}$$

According to the mapping rule, although a sentence may give rise to a number of word pairs, only the pair with the minimum word distance is selected. We do not take into consideration of the other words in a sentence as *relevant opinions* are generally formed in close proximity. A sentence is regarded non-opinionated unless it contains at least one word pair.

In practice, not all word pairs carry equal weights to express a *relevant opinion* as the contribution of an opinion word differs from different target topics, and vice versa. For example, the word pair < t_1, o_2> should be more probable as a *relevant opinion* than < t_1, o_1>. To consider

that, inter-sentence contextual information is explored. This is achieved by assigning a weight to each word pair to measure their associative degrees to different queries. We believe that the more a word pair appears the higher should be the weight between the opinion and the target in the context.

We will describe how to utilize intra-sentence contextual information to express *relevant opinion*, and inter-sentence information to measure the degree of each word pair through a graph-based model in the following section.

3.2 HITS Model

We propose an opinion retrieval model based on HITS, a popular graph ranking algorithm (Kleinberg, 1999). By considering both intra-sentence information and inter-sentence information, we can determine the weight of a word pair and rank the documents.

HITS algorithm distinguishes hubs and authorities in objects. A hub object has links to many authorities. An authority object, which has high-quality content, would have many hubs linking to it. The hub scores and authority scores are computed in an iterative way. Our proposed opinion retrieval model contains two layers. The upper level contains all the topic-sentiment word pairs $p_{ij} = \{< t_i, o_j > | t_i \in V_t, o_j \in V_o)\}$. The lower level contains all the documents to be retrieved. Figure 3 gives the bipartite graph representation of the HITS model.

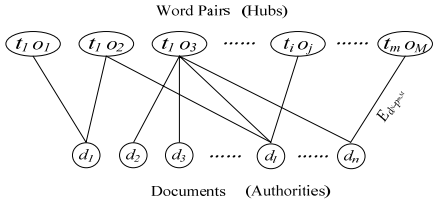


Figure 3: Bipartite link graph.

For our purpose, the word pairs layer is considered as hubs and the documents layer authorities. If a word pair occurs in one sentence of a document, there will be an edge between them. In Figure 3, we can see that the word pair that has links to many documents can be assigned a high weight to denote a strong associative degree between the topic term and a sentiment word, and it likely expresses a *relevant opinion*. On the other hand, if a document has links to many word pairs, the document is with many *relevant opinions*, and it will result in high ranking.

Formally, the representation for the bipartite graph is denoted as $G =< H_p, A_d, E_{dp} >$, where $H_p = \{p_{ij}\}$ is the set of all pairs of topic words and sentiment words, which appear in one sentence. $A_d = \{d_k\}$ is the set of documents. $E_{dp} = \{e_{ij}^k | p_{ij} \in H_p, d_k \in A_d\}$ corresponds to the connection between documents and topic-sentiment word pairs. Each edge e_{ij}^k is associated with a weight $w_{ij}^k \in [0,1]$ denoting the contribution of p_{ij} to the document d_k. The weight w_{ij}^k is computed by the contribution of word pair p_{ij} in all sentences of d_k as follows:

$$w_{ij}^k = \frac{1}{|d_k|} \sum_{p_{ij} \in s_l \in d_k} [\lambda \cdot rel(t_i, s_l) + (1-\lambda) opn(o_j, s_l)] \quad (1)$$

- $|d_k|$ is the number of sentences in d_k;
- λ is introduced as the trade-off parameter to balance the $rel(t_i, s_l)$ and $opn(o_j, s_l)$;
- $rel(t_i, s_l)$ is computed to judge the relevance of t_i in s_l which belongs to d_k;

$$rel(t_i, s_l) = tf_{t_i, s_l} \times isf_{t_i} \quad (2)$$

where tf_{t_i,s_l} is the number of t_i appears in s_l, and

$$isf_{t_i} = \log\left(\frac{N+1}{0.5+sf_{t_i}}\right) \quad (3)$$

where sf_{t_i} is the number of sentences that the word t_i appears in.

- $opn(o_j, s_l)$ is the contribution of o_j in s_l which belongs to d_k.

$$opn(o_j, s_l) = \frac{tf_{o_j, s_l}}{tf_{o_j, s_l} + 0.5 + (1.5 \times \frac{len(s_l)}{asl})} \quad (4)$$

where asl is the average number of sentences in d_k; tf_{t_i,o_j} is the number of o_j appears in s_l (Allan et al., 2003; Otterbacher et al., 2005).

It is found that the contribution of a sentiment word o_j will not decrease even if it appears in all the sentences. Therefore in Equation 4, we just use the length of a sentence instead of isf_{o_j} to normalize long sentences which would likely contain more sentiment words.

The authority score $AuthScore^{(T+1)}(d_k)$ of document d_k and a hub score $HubScore^{(T+1)}(p_{ij})$ of p_{ij} at the $(T+1)^{\text{th}}$ iteration are computed based on the hub scores and authority scores in the T^{th} iteration as follows.

$$AuthScore^{(T+1)}(d_k) = \sum_{p_{ij} \in H_p} w_{ij}^k \times HubScore^T(p_{ij}) \quad (5)$$

$$HubScore^{(T+1)}(p_{ij}) = \sum_{d_k \in A_d} w_{ij}^k \times AuthScore^T(d_k) \quad (6)$$

We let $L = (L_{i,j})_{|H_p| \times |A_d|}$ denote the adjacency matrix.

$$\vec{a}^{(T+1)} = L\vec{h}^{(T)} \quad (7)$$

$$\vec{h}^{(T+1)} = L^T\vec{a}^{(T)} \quad (8)$$

where $\vec{a}^{(T)} = [AuthScore^T(d_k)]_{|A_d| \times 1}$ is the vector of authority scores for documents at the T^{th} iteration and $\vec{h}^{(T)} = [HubScore^T(p_{ij})]_{|H_p| \times 1}$ is the vector of hub scores for the word pairs at T^{th} iteration. In order to ensure convergence of the iterative form, $\vec{a}$ and $\vec{h}$ are normalized in each iteration cycle.

For computation of the final scores, the initial scores of all documents are set to $\frac{1}{\sqrt{n}}$, and topic-sentiment word pairs are set to $\frac{1}{\sqrt{m \times M}}$. The above iterative steps are then used to compute the new scores until convergence. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any nodes falls below a given threshold (Wan et al., 2008; Li et al., 2009; Erkan and Radev, 2004). In our model, we use the hub scores to denote the associative degree of each word pair and the authority scores as the total scores. The documents are then ranked based on the total scores.

4 Experiment

We performed the experiments on the Chinese benchmark dataset to verify our proposed approach for opinion retrieval. We first tested the effect of the parameter λ of our model. To demonstrate the effectiveness of our opinion retrieval model, we compared its performance with the same of other approaches. In addition, we studied each individual query to investigate the influence of query to our model. Furthermore, we showed the top-5 highest weight word pairs of 5 queries to further demonstrate the effect of word pair.

4.1 Experiment Setup

4.1.1 Benchmark Datasets

Our experiments are based on the Chinese benchmark dataset, COAE08 (Zhao et al., 2008). COAE dataset is the benchmark data set for the opinion retrieval track in the Chinese Opinion Analysis Evaluation (COAE) workshop, consisting of blogs and reviews. 20 queries are provided in COAE08. In our experiment, we created relevance judgments through pooling method, where documents are ranked at different levels: irrelevant, relevant but without opinion, and relevant with opinion. Since polarity is not considered, all relevant documents with opinion are classified into the same level.

4.1.2 Sentiment Lexicon

In our experiment, the sentiment lexicon is composed by the following resources (Xu et al., 2007):

(1) *The Lexicon of Chinese Positive Words,* which consists of 5,054 positive words and the *Lexicon of Chinese Negative Words*, which consists of 3,493 negative words;

(2) The opinion word lexicon provided by National Taiwan University which consists of 2,812 positive words and 8,276 negative words;

(3) Sentiment word lexicon and comment word lexicon from *Hownet*. It contains 1836 positive sentiment words, 3,730 positive comments, 1,254 negative sentiment words and 3,116 negative comment words.

The different graphemes corresponding to Traditional Chinese and Simplified Chinese are both considered so that the sentiment lexicons from different sources are applicable to process Simplified Chinese text. The lexicon was manually verified.

4.1.3 Topic Term Collection

In order to acquire the collection of topic terms, we adopt two expansion methods, dictionary-based method and pseudo relevance feedback method.

The dictionary-based method utilizes Wikipedia (Popescu and Etzioni, 2005) to find an entry page for a phrase or a single term in a query. If such an entry exists, all titles of the entry page are extracted as synonyms of the query concept. For example, if we search “绿坝” (Green Tsunami, a firewall) in Wikipedia, it is re-directed to an entry page titled “花季护航” (Youth Escort). This term is then added as a synonym of “绿坝” (Green Tsunami) in the query. Synonyms are treated the same as the original query terms in a retrieval process. The content words in the entry page are ranked by their frequencies in the page. The top-*k* terms are returned as potential expanded topic terms.

The second query expansion method is a web-based method. It is similar to the pseudo relevance feedback expansion but using web documents as the document collection. The query is submitted to a web search engine, such as Google, which returns a ranked list of documents. In the top-*n* documents, the top-*m* topic terms which are highly correlated to the query terms are returned.

4.2 Performance Evaluation

4.2.1 Parameter Tuning

We first studied how the parameter λ (see Equation 1) influenced the mean average precision (MAP) in our model. The result is given in Figure 4.

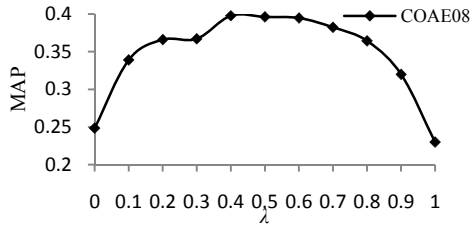


Figure 4: Performance of MAP with varying λ.

Best MAP performance was achieved in COAE08 evaluation, when λ was set between 0.4 and 0.6. Therefore, in the following experiments, we set $\lambda = 0.4$.

4.2.2 Opinion Retrieval Model Comparison

To demonstrate the effectiveness of our proposed model, we compared it with the following models using different evaluation metrics:

(1) IR: We adopted a classical information retrieval model, and further assumed that all retrieved documents contained *relevant opinions*.

(2) Doc: The 2-stage document-based opinion retrieval model was adopted. The model used sentiment lexicon-based method for opinion identification and a conventional information retrieval method for relevance detection.

(3) ROSC: This was the model which achieved the best run in TREC Blog 07. It employed machine learning method to identify opinions for each sentence, and to determine the target topic by a *NEAR* operator.

(4) ROCC: This model was similar to ROSC, but it considered the factor of sentence and regarded the count of relevant opinionated sentence to be the opinion score (Zhang and Yu, 2007). In our experiment, we treated this model as the evaluation baseline.

(5) GORM: our proposed graph-based opinion retrieval model.

Approach	COAE08			
	Evaluation metrics			
Run id	MAP	R-pre	bPref	P@10
IR	0.2797	0.3545	0.2474	0.4868
Doc	0.3316	0.3690	0.3030	0.6696
ROSC	0.3762	0.4321	0.4162	0.7089
Baseline	0.3774	0.4411	0.4198	0.6931
GORM	**0.3978**	**0.4835**	**0.4265**	**0.7309**

Table 1: Comparison of different approaches on COAE08 dataset, and the best is highlighted.

Most of the above models were originally designed for opinion retrieval in English, and re-designed them to handle Chinese opinionated documents. We incorporated our own Chinese sentiment lexicon for this purpose. In our experiments, in addition to MAP, other metrics such as R-precision (R-prec), binary Preference (bPref) and Precision at 10 documents (P@10) were also used. The evaluation results based on these metrics are shown in Table 1.

Table 1 summarized the results obtained. We found that GORM achieved the best performance in all the evaluation metrics. Our baseline, ROSC and GORM which were sentence-based approaches achieved better performance than the document-based approaches by 20% in average. Moreover, our GORM approach did not use machine learning techniques, but it could still achieve outstanding performance.

To study GORM influenced by different queries, the MAP from median average precision on individual topic was shown in Figure 5.

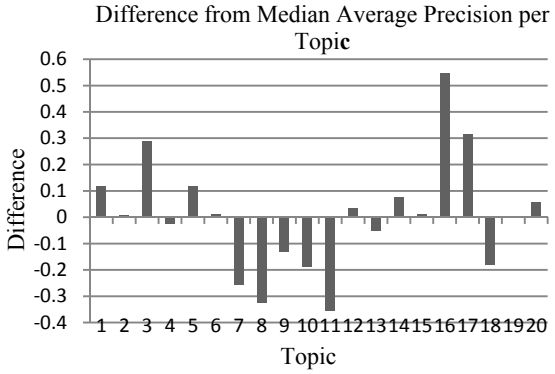


Figure 5: Difference of MAP from Median on COAE08 dataset. (MAP of Median is 0.3724)

As shown in Figure 5, the MAP performance was very low on topic 8 and topic 11. Topic 8, i.e. '成龙' (Jackie Chan), it was influenced by topic 7, i.e. '李连杰' (Jet Lee) as there were a number of similar relevant targets for the two topics, and therefore many word pairs ended up the same. As a result, documents belonging to topic 7 and topic 8 could not be differentiated, and they both performed badly. In order to solve this problem, we extracted the topic term with highest relevant weight in the sentence to form word pairs so that it reduce the impact on the topic terms in common. 24% and 30% improvement were achieved, respectively.

As to topic 11, i.e. '指环王' (Lord of King), there were only 8 relevant documents without any opinion and 14 documents with *relevant opinions*. As a result, the graph constructed by insufficient documents worked ineffectively.

Except for the above queries, GORM performed well in most of the others. To further investigate the effect of word pair, we summarized the top-5 word pairs with highest weight of 5 queries in Table 2.

Top-5 MAP				
陈凯歌 Chen Kaige	国六条 Six States	宏观调控 Macro-regulation	周星驰 Stephen Chow	Vista Vista
<陈凯歌 支持> Chen Kaige Support	<房价 上涨> Room rate Rise	<经济 平稳> Economics Steady	<电影 喜欢> Movie Like	<价格 贵> Price Expensive
<陈凯歌 最佳> Chen Kaige Best	<调控 加强> Regulate Strengthen	<价格 上涨> Price Rise	<周星驰 喜欢> Stephen Chow Like	<微软 喜欢> Microsoft Like
<《无极》 骂> Limitless Revile	<中央 加强> CCP Strengthen	<发展 平稳> Development Steady	<主角 最佳> Protagonist Best	<Vista 推荐> Vista Recommend
<影片 优秀> Movie Excellent	<房价 平稳> Room rate Steady	<消费 上涨> Consume Rise	<喜剧 好> Comedy Good	<问题 重要> Problem Vital
<阵容 强大的> Cast Strong	<住房 保障> Housing Security	<社会 保障> Social Security	<作品 精彩> Works Splendid	<性能 不> Performance No

Table 2: Top-5 highest weight word pairs for 5 queries in COAE08 dataset.

Table 2 showed that most word pairs could represent the *relevant opinions* about the corresponding queries. This showed that inter-sentence information was very helpful to identify the associative degree of a word pair. Furthermore, since word pairs can indicate relevant opinions effectively, it is worth further study on how they could be applied to other opinion oriented applications, e.g. opinion summarization, opinion prediction, etc.

5 Related Work

Our research focuses on *relevant opinion* rather than on relevant document retrieval. We, therefore, review related works in opinion identification research. Furthermore, we do not support the conventional 2-stage opinion retrieval approach. We conducted literature review on unified opinion retrieval models and related work in this area is presented in the section.

5.1 Lexicon-based Opinion Identification

Different from traditional IR, opinion retrieval focuses on the opinion nature of documents. During the last three years, NTICR and TREC evaluations have shown that sentiment lexicon-based methods led to good performance in opinion identification.

A lightweight lexicon-based statistical approach was proposed by Hannah et al. (2007). In this method, the distribution of terms in relevant opinionated documents was compared to their distribution in relevant fact-based documents to calculate an opinion weight. These weights were used to compute opinion scores for each retrieved document. A weighted dictionary was generated from previous TREC relevance data (Amati et al., 2007). This dictionary was submitted as a query to a search engine to get an initial query-independent opinion score of all retrieved documents. Similarly, a pseudo opinionated word composed of all opinion words was first created, and then used to estimate the opinion score of a document (Na et al., 2009). This method was shown to be very effective in TREC evaluations (Lee et al., 2008). More recently, Huang and Croft (2009) proposed an effective relevance model, which integrated both query-independent and query-dependent sentiment words into a mixture model.

In our approach, we also adopt sentiment lexicon-based method for opinion identification. Unlike the above methods, we generate a weight to a sentiment word for each target (associated topic term) rather than assign a unified weight or an equal weight to the sentiment word for the whole topics. Besides, in our model no training data is required. We just utilize the structure of our graph to generate a weight to reflect the associative degree between the two elements of a word pair in different context.

5.2 Unified Opinion Retrieval Model

In addition to conventional 2-stage approach, there has been some research on unified opinion retrieval models.

Eguchi and Lavrenko proposed an opinion retrieval model in the framework of generative language modeling (Eguchi and Lavrenko, 2006). They modeled a collection of natural language documents or statements, each of which consisted of some topic-bearing and some sentiment-bearing words. The sentiment was either represented by a group of predefined seed words, or extracted from a training sentiment corpus. This model was shown to be effective on the MPQA corpus.

Mei et al. tried to build a fine-grained opinion retrieval system for consumer products (Mei et al., 2007). The opinion score for a product was a mixture of several facets. Due to the difficulty in

associating sentiment with products and facets, the system was only tested using small scale text collections.

Zhang and Ye proposed a generative model to unify topic relevance and opinion generation (Zhang and Ye, 2008). This model led to satisfactory performance, but an intensive computation load was inevitable during retrieval, since for each possible candidate document, an opinion score was summed up from the generative probability of thousands of sentiment words.

Huang and Croft proposed a unified opinion retrieval model according to the Kullback-Leibler divergence between the two probability distributions of opinion relevance model and document model (Huang and Croft, 2009). They divided the sentiment words into query-dependent and query-independent by utilizing several sentiment expansion techniques, and integrated them into a mixed model. However, in this model, the contribution of a sentiment word was its corresponding incremental mean average precision value. This method required that large amount of training data and manual labeling.

Different from the above opinion retrieval approaches, our proposed graph-based model processes opinion retrieval in the granularity of sentence. Instead of *bag-of-word*, the sentence is split into word pairs which can maintain the contextual information. On the one hand, word pair can identify the *relevant opinion* according to intra-sentence contextual information. On the other hand, it can measure the degree of a *relevant opinion* by considering the inter-sentence contextual information.

6 Conclusion and Future Work

In this work we focus on the problem of opinion retrieval. Different from existing approaches, which regard document relevance as the key indicator of opinion relevance, we propose to explore the relevance of individual opinion. To do that, opinion retrieval is performed in the granularity of sentence. We define the notion of word pair, which can not only maintain the association between the opinion and the corresponding target in the sentence, but it can also build up the relationship among sentences through the same word pair. Furthermore, we convert the relationships between word pairs and sentences into a unified graph, and use the HITS algorithm to achieve document ranking for opinion retrieval. Finally, we compare our approach with existing methods. Experimental results show that our proposed model performs well on COAE08 dataset.

The novelty of our work lies in using word pairs to represent the information needs for opinion retrieval. On the one hand, word pairs can identify the *relevant opinion* according to intra-sentence contextual information. On the other hand, word pairs can measure the degree of a *relevant opinion* by taking inter-sentence contextual information into consideration. With the help of word pairs, the information needs for opinion retrieval can be represented appropriately.

In the future, more research is required in the following directions:

(1) Since word pairs can indicate relevant opinions effectively, it is worth further study on how they could be applied to other opinion oriented applications, e.g. opinion summarization, opinion prediction, etc.

(2) The characteristics of blogs will be taken into consideration, i.e., the post time, which could be helpful to create a more time sensitivity graph to filter out fake opinions.

(3) Opinion holder is another important role of an opinion, and the identification of opinion holder is a main task in NTCIR. It would be interesting to study opinion holders, e.g. its seniority, for opinion retrieval.

Acknowledgements: This work is partially supported by the Innovation and Technology Fund of Hong Kong SAR (No. ITS/182/08) and National 863 program (No. 2009AA01Z150). Special thanks to Xu Hongbo for providing the Chinese sentiment resources. We also thank Bo Chen, Wei Gao, Xu Han and anonymous reviewers for their helpful comments.

References

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 314-321. ACM.

Giambattista Amati, Edgardo Ambrosi, Marco Bianchi, Carlo Gaibisso, and Giorgio Gambosi. 2007. FUB, IASI-CNR and University of Tor Vergata at TREC 2007 Blog Track. In *Proceedings of the 15th Text Retrieval Conference*.

Koji Eguchi and Victor Lavrenko. Sentiment retrieval using generative models. 2006. In *EMNLP '06, Proceedings of 2006 Conference on Empirical Methods in Natural Language Processing*, page 345-354.

Gunes Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP '04, Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*.

David Hannah, Craig Macdonald, Jie Peng, Ben He, and Iadh Ounis. 2007. University of Glasgow at TREC 2007: Experiments in Blog and Enterprise Tracks with Terrier. In *Proceedings of the 15th Text Retrieval Conference*.

Xuanjing Huang, William Bruce Croft. 2009. A Unified Relevance Model for Opinion Retrieval. In *Proceedings of CIKM*.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM, 46(5)*: 604-632.

Yeha Lee, Seung-Hoon Na, Jungi Kim, Sang-Hyob Nam, Hun-young Jung, Jong-Hyeok Lee. 2008. KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. In *Proceedings of the 15th Text Retrieval Conference*.

Fangtao Li, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering Opinion Questions with Random Walks on Graphs. In *ACL '09, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.*

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinion s on the web. In *WWW '05: Proceedings of the 14th International Conference on World Wide Web*.

Craig Macdonald and Iadh Ounis. 2007. Overview of the TREC-2007 Blog Track. In *Proceedings of the 15th Text Retrieval Conference*.

Craig Macdonald and Iadh Ounis. 2006. Overview of the TREC-2006 Blog Track. In *Proceedings of the 14th Text Retrieval Conference*.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and Chengxiang Zhai. 2007. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *WWW '07: Proceedings of the 16 International Conference on World Wide Web*.

Seung-Hoon Na, Yeha Lee, Sang-Hyob Nam, and Jong-Hyeok Lee. 2009. Improving opinion retrieval based on query-specific sentiment lexicon. In *ECIR '09: Proceedings of the 31st annual European Conference on Information Retrieval*, pages 734-738.

Douglas Oard, Tamer Elsayed, Jianqiang Wang, Yejun Wu, Pengyi Zhang, Eileen Abels, Jimmy Lin, and Dagbert Soergel. 2006. TREC-2006 at Maryland: Blog, Enterprise, Legal and QA Tracks. In *Proceedings of the 15th Text Retrieval Conference*.

Jahna Otterbacher, Gunes Erkan, and Dragomir R. Radev. 2005. Using random walks for question-focused sentence retrieval. In *EMNLP '05, Proceedings of 2005 Conference on Empirical Methods in Natural Language Processing*.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2): 1-135.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinion s from reviews. In *EMNLP '05, Proceedings of 2005 Conference on Empirical Methods in Natural Language Processing*.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *SIGIR '08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299-306. ACM.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP '05, Proceedings of 2005 Conference on Empirical Methods in Natural Language Processing*.

Ruifeng Xu, Kam-Fai Wong and Yunqing Xia. 2007. Opinmine - Opinion Analysis System by CUHK for NTCIR-6 Pilot Task. In *Proc*eedings *of NTCIR-6*.

Min Zhang and Xingyao Ye. 2008. A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 411-418. ACM.

Wei Zhang and Clement Yu. 2007. UIC at TREC 2007 Blog Track. In *Proceedings of the 15th Text Retrieval Conference*.

Jun Zhao, Hongbo Xu, Xuanjing Huang, Songbo Tan, Kang Liu, and Qi Zhang. 2008. Overview of Chinese Opinion Analysis Evaluation 2008. In *Proceedings of the First Chinese Opinion Analysis Evaluation*.

Generating Fine-Grained Reviews of Songs From Album Reviews

Swati Tata and **Barbara Di Eugenio**
Computer Science Department
University of Illinois, Chicago, IL, USA
{stata2 | bdieugen}@uic.edu

Abstract

Music Recommendation Systems often recommend individual songs, as opposed to entire albums. The challenge is to generate reviews for each song, since only full album reviews are available on-line. We developed a summarizer that combines information extraction and generation techniques to produce summaries of reviews of individual songs. We present an intrinsic evaluation of the extraction components, and of the informativeness of the summaries; and a user study of the impact of the song review summaries on users' decision making processes. Users were able to make quicker and more informed decisions when presented with the summary as compared to the full album review.

1 Introduction

In recent years, the personal music collection of many individuals has significantly grown due to the availability of portable devices like MP3 players and of internet services. Music listeners are now looking for techniques to help them manage their music collections and explore songs they may not even know they have (Clema, 2006). Currently, most of those electronic devices follow a Universal Plug and Play (UPNP) protocol (UPN, 2008), and can be used in a simple network, on which the songs listened to can be monitored. Our interest is in developing a Music Recommendation System (Music RS) for such a network.

Commercial web-sites such as Amazon (www.amazon.com) and Barnes and Nobles (www.bnn.com) have deployed Product Recommendation Systems (Product RS) to help customers choose from large catalogues of products. Most Product RSs include reviews from customers who bought or tried the product. As the number of reviews available for each individual product increases, RSs may overwhelm the user if they make all those reviews available. Additionally, in some reviews only few sentences actually describe the recommended product, hence, the interest in opinion mining and in summarizing those reviews.

A Music RS could be developed along the lines of Product RSs. However, Music RSs recommend individual tracks, not full albums, e.g. see www.itunes.com. Summarizing reviews becomes more complex: available data consists of album reviews, not individual song reviews (www.amazon.com, www.epinions.com). Comments about a given song are fragmented all over an album review. Though some web-sites like www.last.fm allow users to comment on individual songs, the comments are too short (a few words such as "awesome song") to be counted as a full review.

In this paper, after presenting related work and contrasting it to our goals in Section 2, we discuss our prototype Music RS in Section 3. We devote Section 4 to our summarizer, that extracts comments on individual tracks from album reviews and produces a summary of those comments for each individual track recommended to the user. In Section 5, we report two types of evaluation: an intrinsic evaluation of the extraction components, and of the coverage of the summary; an extrinsic evaluation via a between-subject study. We found that users make quicker and more informed decisions when presented with the song review summaries as opposed to the full album review.

2 Related Work

Over the last decade, summarization has become a hot topic for research. Quite a few systems were developed for different tasks, including multi-document summarization (Barzilay and McKeown, 2005; Soubbotin and Soubbotin, 2005; Nastase, 2008).

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1376–1385,
Uppsala, Sweden, 11-16 July 2010.

What's not to get? Yes, **Maxwell**, and **Octopus** are a bit silly! ...
.....
"Something" and **"Here Comes The Sun"** are two of George's best songs ever (and **"Something"** may be the single greatest love song ever). **"Oh Darling"** is a bluesy masterpiece with Paul screaming.....
.....
"Come Together" contains a great riff, but he ended up getting sued over the lyrics by Chuck Berry.....

Figure 1: A sample review for the album "Abbey Road"

Whereas summarizing customer reviews can be seen as multi-document summarization, an added necessary step is to first extract the most important features customers focus on. Hence, summarizing customer reviews has mostly been studied as a combination of machine learning and NLP techniques (Hu and Liu, 2004; Gamon et al., 2005). For example, (Hu and Liu, 2004) use associative mining techniques to identify features that frequently occur in reviews taken from `www.epinions.com` and `www.amazon.com`. Then, features are paired to the nearest words that express some opinion on that feature. Most work on product reviews focuses on identifying sentences and polarity of opinion terms, not on generating a coherent summary from the extracted features, which is the main goal of our research. Exceptions are (Carenini et al., 2006; Higashinaka et al., 2006), whose focus was on extracting domain specific ontologies in order to structure summarization of customer reviews.

Summarizing reviews on objects different from products, such as restaurants (Nguyen et al., 2007), or movies (Zhuang et al., 2006), has also been tackled, although not as extensively. We are aware of only one piece of work that focuses on music reviews (Downie and Hu, 2006). This study is mainly concerned with identifying descriptive patterns in positive or negative reviews but not on summarizing the reviews.

2.1 Summarizing song reviews is different

As mentioned earlier, using album reviews for song summarization poses new challenges:
a) Comments on features of a song are embedded and fragmented within the album reviews, as shown in Figure 1. It is necessary to correctly map features to songs.
b) Each song needs to be identified each time it is referred to in the review. Titles are often abbreviated, and in different ways, even in the same review – e.g. see *Octopus* for *Octopus's Garden* in Figure 1. Additionally, song titles need not be noun phrases and hence NP extraction algorithms miss many occurrences, as was shown by preliminary experiments we ran.
c) Reviewers focus on both inherent features such as lyrics, genre and instruments, but also on people (artist, lyricist, producer etc.), unlike in product reviews where manufacturer/designer are rarely mentioned. This variety of features makes it harder to generate a coherent summary.

3 SongRecommend: Prototype Music RS

Figure 2 shows the interface of our prototype Music RS. It is a simple interface dictated by our focus on the summarization process (but it was informed by a small pilot study). Moving from window to window and from top to bottom:
a) The top leftmost window shows different devices on which the user listens to songs. These devices are monitored with a UPNP control point. Based on the messages received by the control point, the user activities, including the metadata of the song, are logged.
b) Once the user chooses a certain song on one of the devices (see second window on top), we display more information about the song (third top window); we also identify related songs from the internet, including: other songs from the same album, popular songs of the artist and popular songs of related artists, as obtained from Yahoo Music.
c) The top 25 recommendations are shown in the fourth top window. We use the SimpleKMeans Clustering (Mitchell, 1997) to identify and rank the top twenty-five songs which belong to the same cluster and are closest to the given song. Closeness between two songs in a cluster is measured as the number of attributes (album, artist etc) of the songs that match.
d) When the user clicks on *More Info* for one of the recommended songs, the pop-up, bottom window is displayed, which contains the summary of the reviews for the specific song.

4 Extraction and Summarization

Our summarization framework consists of the five tasks illustrated in Figure 3. The first two tasks pertain to information extraction, the last three to repackaging the information and generating a co-

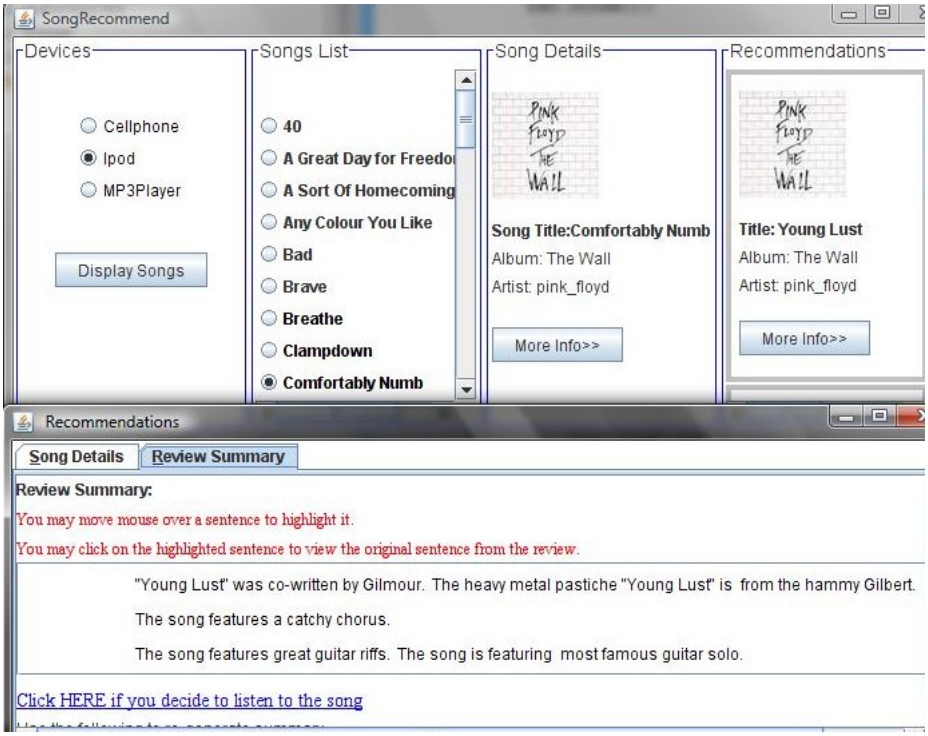


Figure 2: SongRecommend Interface

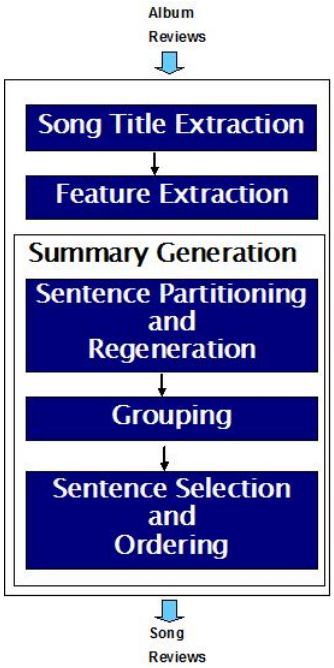


Figure 3: Summarization Pipeline

herent summary. Whereas the techniques we use for each individual step are state-of-the-art, our approach is innovative in that it integrates them into an effective end-to-end system. Its effectiveness is shown by the promising results obtained both via the intrinsic evaluation, and the user study. Our framework can be applied to any domain where reviews of individual components need to be summarized from reviews of collections, such as reviews of different hotels and restaurants in a city.

Our corpus was opportunistically collected from `www.amazon.com` and `www.epinions.com`. It consists of 1350 album reviews across 27 albums (50 reviews per album). 50 randomly chosen reviews were used for development. Reviews have noise, since the writing is informal. We did not clean it, for example we did not correct spelling mistakes. This corpus was annotated for song titles and song features. Feature annotation consists of marking a phrase as a feature and matching it with the song to which the feature is attributed. Note that we have no a priori inventory of features; what counts as features of songs emerged from the annotation, since annotators were asked to annotate for noun phrases which contain "any song related term or terms spoken in the context of a song". Further, they were given about 5 positive and 5 negative

What's not to get? Yes, <**song id=3**>**Maxwell**</**song**>, and <**song id=5**>**Octopus**</**song**> are a bit silly! ...
.....
.....
<**song id=2**>**"Something"**</**song**> and <**song id=7**>**"Here Comes The Sun"**</**song**> are two of <**feature id=(2,7)**>**George's**</**feature**> best songs ever (and <**song id=2**>**"Something"**</**song**> may be
.....
<**song id=4**>**"Oh Darling"**</**song**> is a <**feature id=4**>**bluesy masterpiece**</**feature**> with <**feature id=4**>**Paul**</**feature**> screaming.....
.....
<**song id=1**>**"Come Together"**</**song**> contains a great <**feature id=1**>**riff**</**feature**>, but ...

Figure 4: A sample annotated review

examples of features. Figure 4 shows annotations for the excerpt in Figure 1. For example in Figure 4, *George, Paul, bluesy masterpiece* and *riff* have been marked as features. Ten randomly chosen reviews were doubly annotated for song titles and features. The Kappa co-efficient of agreement on both was excellent (**0.9**), hence the rest of the corpus was annotated by one annotator only. The two annotators were considered to be in agreement on a feature if they marked the same head of phrase **and** attributed it to the same song.

We will now turn to describing the component tasks. The algorithms are described in full in (Tata, 2010).

4.1 Title Extraction

Song identification is the first step towards summarization of reviews. We identify a string of words as the title of a song to be extracted from an album review if it (1) includes some or all the words in the title of a track of that album, and (2) this string occurs in the right context. Constraint (2) is necessary because the string of words corresponding to the title may appear in the lyrics of the song or anywhere else in the review. The string *Maxwell's Silver Hammer* counts as a title only in sentence (a) below; the second sentence is a verse in the lyrics:

a. Then, the wild and weird "Maxwell's Silver Hammer."

b. Bang, Bang, maxwell's silver hammer cam down on her head.

Similar to Named Entity Recognition (Schedl et al., 2007), our approach to song title extraction is based on *n-grams*. We proceed album by album. Given the reviews for an album and the list of songs in that album, first, we build a lexicon of all the words in the song titles. We also segment the reviews into sentences via sentence boundary detection. All 1,2,3,4-grams for each sentence (the upper-bound 4 was determined experimentally) in the review are generated. First, n-grams that contain at least one word with an edit distance greater than one from a word in the lexicon are filtered out. Second, if higher and lower order n-grams overlap at the same position in the same sentence, lower order n-grams are filtered out. Third, the n-grams are merged if they occur sequentially in a sentence. Fourth, the n-grams are further filtered to include only those where (i) the n-gram is within quotation marks; and/or (ii) the first character of each word in the n-gram is upper case. This filters n-grams such as those shown in sentence (b) above. All the n-grams remaining at this point are potential song titles. Finally, for each n-gram, we retrieve the set of IDs for each of its words and intersect those sets. This intersection always resulted in one single song ID, since song titles in each album differ by at least one content word. Recall that the algorithm is run on reviews for each album separately.

4.2 Feature Extraction

Once the song titles are identified in the album review, sentences with song titles are used as anchors to (1) identify *segments* of texts that talk about a specific song, and then (2) extract the feature(s) that the pertinent text segment discusses.

The first step roughly corresponds to identifying the flow of topics in a review. The second step corresponds to identifying the properties of each song. Both steps would greatly benefit from reference resolution, but current algorithms still have a low accuracy. We devised an approach that combines text tiling (Hearst, 1994) and domain heuristics. The text tiling algorithm divides the text into coherent discourse units, to describe the sub-topic structure of the given text. We found the relatively coarse segments the text tiling algorithm provides sufficient to identify different topics.

An album review is first divided into segments using the text tiling algorithm. Let $[seg_1, seg_2, ..., seg_k]$ be the segments obtained. The segments that contain potential features of a song are identified using the following heuristics:

Step 1: Include seg_i if it contains a song title.

These segments are more likely to contain features of songs as they are composed of the sentences surrounding the song title.

Step 2: Include seg_{i+1} if seg_i is included and seg_{i+1} contains one or more *feature terms*.

Since we have no a priori inventory of features (the feature annotation will be used for evaluation, not for development), we use WordNet (Fellbaum, 1998) to identify *feature terms*: i.e., those nouns whose synonyms, direct hypernym or direct hyponym, or the definitions of any of those, contain the terms "music" or "song", or any form of these words like "musical", "songs" etc, for at least one sense of the noun. Feature terms exclude the words "music", "song", the artist/band/album name as they are likely to occur across album reviews. All feature terms in the final set of segments selected by the heuristics are taken to be features of the song described by that segment.

4.3 Sentence Partitioning and Regeneration

After extracting the sentences containing the features, the next step is to divide the sentences into two or more "sub-sentences", if necessary. For example, *"McCartney's bouncy bass-line is especially wonderful, and George comes in with an excellent, minimal guitar solo."* discusses both features *bass* and *guitar*. Only a portion of the sentence describes the *guitar*. This sentence can thus be divided into two individual sentences. Removing parts of sentences that describe another feature, will have no effect on the summary as a whole as the portions that are removed will be present in the group of sentences that describe the other feature.

To derive n sentences, each concerning a single feature f, from the original sentence that covered n features, we need to:

1. Identify portions of sentences relevant to each feature f (partitioning)
2. Regenerate each portion as an independent sentence, which we call f-sentence.

To identify portions of the sentence relevant to the single feature f, we use the Stanford Typed Dependency Parser (Klein and Manning, 2002; de Marnee and Manning, 2008). Typed Dependencies describe grammatical relationships between pairs of words in a sentence. Starting from the feature term f in question, we collect all the nouns, adjectives and verbs that are directly related to it in the sentence. These nouns, adjectives and verbs become the components of the new f-sentence. Next, we need to adjust their number and forms. This is a natural language generation task, specifically, sentence realization.

1. **"Maxwell"** is a bit silly.
2. **"Octopus"** is a bit silly.
3. **"Something"** is George's best song.
4. **"Here Comes The Sun"** is George's best song.
5. **"Something"** may be the single greatest love song.
6. **"Oh! Darling"** is a bluesy masterpiece.
7. **"Come Together"** contains a great riff.

Figure 5: f-sentences corresponding to Figure 1

We use YAG (McRoy et al., 2003), a template based sentence realizer. *clause* is the main template used to generate a sentence. Slots in a template can in turn be templates. The grammatical relationships obtained from the Typed Dependency Parser such as *subject* and *object* identify the slots and the template the slots follows; the words in the relationship fill the slot. We use a morphological tool (Minnen et al., 2000) to obtain the base form from the original verb or noun, so that YAG can generate grammatical sentences. Figure 5 shows the regenerated review from Figure 1.

YAG regenerates as many f-sentences from the original sentence, as many features were contained in it. By the end of this step, for each feature f of a certain song s_i, we have generated a set of f-sentences. This set also contains every original sentence that only covered the single feature f.

4.4 Grouping

f-sentences are further grouped, by sub-feature and by polarity. As concerns sub-feature grouping, consider the following f-sentences for the feature *guitar*:

a. *George comes in with an excellent, minimal guitar solo.*

b. *McCartney laid down the guitar lead for this track.*

c. *Identical lead guitar provide the rhythmic basis for this song.*

The first sentence talks about the *guitar solo*, the second and the third about the *lead guitar*. This step will create two subgroups, with sentence *a* in one group and sentences *b* and *c* in another. We

Let $[f_x\text{-}s_1, f_x\text{-}s_2, ...f_x\text{-}s_n]$ be the set of sentences for feature f_x and song S_y

Step 1: *Find the longest common n-gram (LCN) between f_x-s_i and f_x-s_j for all $i \neq j$: LCN(f_x-s_i, f_x-s_j)*

Step 2: *If LCN(f_x-s_i, f_x-s_j) contains the feature term and is not the feature term alone, f_x-s_i and f_x-s_j are in the same group.*

Step 3: *For any f_x-s_i, if LCN(f_x-s_i, f_x-s_j) for all i and j, is the feature term, then f_x-s_i belongs to the default group for the feature.*

Figure 6: Grouping sentences by sub-features

Example: *The lyrics are the best*
Adjectives in the sentence: *best*

Senti-wordnet Scores of *best*:
Sense 1 (frequency=2):
positive = 0.625, negative =0 , objective = 0.375

Sense 2 (frequency=1):
positive = 0.75, negative = 0, objective = 0.25

Polarity Scores Calculation:
positive(best) = $\frac{2*0.625+1*0.75}{(2+1)}$ = 0.67
negative(best) = $\frac{2*0+1*0}{(2+1)}$ = 0
objective(best) = $\frac{2*0.375+1*0.25}{(2+1)}$ = 0.33

Since the sentence contains only the adjective *best*, its polarity is positive, from:
Max (positive(best), negative(best), objective(best))

Figure 7: Polarity Calculation

identify subgroups via common n-grams between f-sentences, and make sure that only n-grams that are related to feature f are identified at this stage, as detailed in Figure 6. When the procedure described in Figure 6 is applied to the three sentences above, it identifies *guitar* as the longest pertinent LCN between *a* and *b*, and between *a* and *c*; and *guitar lead* between *b* and *c* (we do not take into account linear order within n-grams, hence *guitar lead* and *lead guitar* are considered identical). Step 2 in Figure 6 will group *b* and *c* together since *guitar lead* properly contains the feature term *guitar*. In Step 3, sentence *a* is sentence f_x-s_i such that its LCN with all other sentences (*b* and *c*) contains only the feature term; hence, sentence *a* is left on its own. Note that Steps 2 and 3 ensure that, among all the possible LNCs between pair of sentences, we only consider the ones containing the feature in question.

As concerns polarity grouping, different reviews may express different opinions regarding a particular feature. To generate a coherent summary that mentions conflicting opinions, we need to subdivide f-sentences according to polarity.

We use SentiWordNet (Esuli and Sebastiani, 2006), an extension of WordNet where each sense of a word is augmented with the probability of that sense being positive, negative or neutral. The overall sentence score is based on the scores of the adjectives contained in the sentence.

Since there are a number of senses for each word, an adjective a_i in a sentence is scored as the normalized weighted scores of each sense of the adjective. For each a_i, we compute three scores, positive, as shown in Formula 1, negative and objective, which are computed analogously:

$$pos(a_i) = \frac{freq_1 * pos_1 + ... + freq_n * pos_n}{(freq_1 + ... + freq_n)} \quad (1)$$

a_i is the i^{th} adjective, $freq_j$ is the frequency of the j^{th} sense of a_i as given by Wordnet, and pos_j is the positive score of the j^{th} sense of a_i, as given by SentiWordnet. Figure 7 shows an example of calculating the polarity of a sentence.

For an f-sentence, three scores will be computed, as the sum of the corresponding scores (positive, negative, objective) of all the adjectives in the sentence. The polarity of the sentence is determined by the maximum of these three scores.

4.5 Selection and Ordering

Finally, the generation of a coherent summary involves selection of the sentences to be included, and ordering them in a coherent fashion. This step has in input groups of f-sentences, where each group pertains to the feature f, one of its subfeatures, and one polarity type (positive, negative, objective). We need to select one sentence from each subgroup to make sure that all essential concepts are included in the summary. Note that if there are contrasting opinions on one feature or subfeatures, one sentence per polarity will be extracted, resulting in potentially inconsistent opinions on that feature to be included in the review (we did not observe this happening frequently, and even if it did, it did not appear to confuse our users).

Recall that at this point, most f-sentences have been regenerated from portions of original sen-

tences (see Section 4.3). Each f-sentence in a subgroup is assigned a score which is equivalent to the number of features in the original sentence from which the f-sentence was obtained. The sentence which has the lowest score in each subgroup is chosen as the representative for that subgroup. If multiple sentences have the lowest score, one sentence is selected randomly. Our assumption is that among the original sentences, a sentence that talks about one feature only is likely to express a stronger opinion about that feature than a sentence in which other features are present.

We order the sentences by exploiting a music ontology (Giasson and Raimond, 2007). We have extended this ontology to include few additional concepts that correspond to features identified in our corpus. Also, we extended each of the classes by adding the domain to which it belongs. We identified a total of 20 different domains for all the features. For example, *[saxophone,drums]* belongs to the domain *Instrument*, and *[tone, vocals]* belong to the domain *Sound*. We also identified the priority order in which each of these domains should appear in the final summary. The ordering of the domains is such that first we present the general features of the song (e.g. *Song*) domain, then present more specific domains (e.g. *Sound, Instrument*). $f-$sentences of a single domain form one paragraph in the final summary. However, features domains that are considered as sub-domains of another domain are included in the same paragraph, but are ordered next to the features of the parent domain. The complete list of domains is described in (Tata, 2010). f-sentences are grouped and ordered according to the domain of the features. Figure 8 shows a sample summary when the extracted sentences are ordered via this method.

> "The Song That Jane Likes" is cute. The song has some nice riffs by Leroi Moore. "The Song That Jane Likes" is also amazing funk number.
>
> The lyrics are sweet and loving.
>
> The song carries a light-hearted tone. It has a catchy tune. The song features some nice accents.
>
> "The Song That Jane Likes" is beautiful song with great rhythm. The funky beat will surely make a move.
>
> It is a heavily acoustic guitar-based song.

Figure 8: Sample summary

5 Evaluation

In this section we report three evaluations, two intrinsic and one extrinsic: evaluation of the song title and feature extraction steps; evaluation of the informativeness of summaries; and a user study to judge how summaries affect decision making.

5.1 Song Title and Feature Extraction

The song title extraction and feature extraction algorithms (Sections 4.1 and 4.2) were manually evaluated on 100 reviews randomly taken from the corpus (2 or 3 from each album). This relatively small number is due to the need to conduct the evaluation manually. The 100 reviews contained 1304 occurrences of song titles and 898 occurrences of song features, as previously annotated.

1294 occurrences of song titles were correctly identified; additionally, 123 spurious occurrences were also identified. This results in a precision of 91.3%, and recall of 98%. The 10 occurrences that were not identified contained either abbreviations like *Dr.* for *Doctor* or spelling mistakes (recall that we don't clean up mistakes).

Of the 898 occurrences of song features, 853 were correctly identified by our feature extraction algorithm, with an additional 41 spurious occurrences. This results in a precision of 95.4% and a recall of 94.9%. Note that a feature (NP) is considered as correctly identified, if its head noun is annotated in a review for the song with correct ID.

As a baseline comparison, we implemented the feature extraction algorithm from (Hu and Liu, 2004). We compared their algorithm to ours on 10 randomly chosen reviews from our corpus, for a total of about 500 sentences. Its accuracy (40.8% precision, and 64.5% recall) is much lower than ours, and than their original results on product reviews (72% precision, and 80% recall).

5.2 Informativeness of the summaries

To evaluate the information captured in the summary, we randomly selected 5 or 6 songs from 10 albums, and generated the corresponding 52 summaries, one per song – this corresponds to a test set of about 500 album reviews (each album has about 50 reviews). Most summary evaluation schemes, for example the Pyramid method (Harnly et al., 2005), make use of reference summaries written by humans. We approximate those gold-standard reference summaries with 2 or 3 critic reviews per album taken from `www.pitchfork.`

com, www.rollingstone.com and www.allmusic.com.

First, we manually annotated both critic reviews and the automatically generated summaries for song titles and song features. 302, i.e., 91.2% of the features identified in the critic reviews are also identified in the summaries (recall that a feature is considered as identified, if the head-noun of the NP is identified by both the critic review and the summary, and attributed to the same song). 64 additional features were identified, for a recall of 82%. It is not surprising that additional features may appear in the summaries: even if only one of the 50 album reviews talks about that feature, it is included in the summary. Potentially, a threshold on frequency of feature mention could increase recall, but we found out that even a threshold of two significantly affects precision.

In a second evaluation, we used our Feature Extraction algorithm to extract features from the critic reviews, for each song whose summary needs to be evaluated. This is an indirect evaluation of that algorithm, in that it shows it is not affected by somewhat different data, since the critic reviews are more formally written. 375, or 95% of the features identified in the critic reviews are also identified in the summaries. 55 additional features were additionally identified, for a recall of 87.5%. These values are comparable, even if slightly higher, to the precision and recall of the manual annotation described above.

5.3 Between-Subject User Study

Our intrinsic evaluation gives satisfactory results. However, we believe the ultimate measure of such a summarization algorithm is an end-to-end evaluation to ascertain whether it affects user behavior, and how. We conducted a between-subject user study, where users were presented with two different versions of our Music RS. For each of the recommended songs, the baseline version provides *only* whole album reviews, the experimental version provides the automatically generated song feature summary, as shown in Figure 2. The interface for the baseline version is similar, but the summary in the bottom window is replaced by the corresponding album review. The presented review is the one among the 50 reviews for that album whose length is closest to the average length of album reviews in the corpus (478 words).

Each user was presented with 5 songs in succession, with 3 recommendations each (only the top 3 recommendations were presented among the available 25, see Section 3). Users were asked to select at least one recommendation for each song, namely, to click on the url where they can listen to the song. They were also asked to base their selection on the information provided by the interface. The first song was a test song for users to get acquainted with the system. We collected comprehensive timed logs of the user actions, including clicks, when windows are open and closed, etc. After using the system, users were administered a brief questionnaire which included questions on a 5-point Likert Scale. 18 users interacted with the baseline version and 21 users with the experimental version (five additional subjects were run but their log data was not properly saved). All users were students at our University, and most of them, graduate students (no differences were found due to gender, previous knowledge of music, or education level).

Our main measure is time on task, the total time taken to select the recommendations from song 2 to song 5 – this excludes the time spent listening to the songs. A t-test showed that users in the experimental version take less time to make their decision when compared to baseline subjects ($p = 0.019$, $t = 2.510$). This is a positive result, because decreasing time to selection is important, given that music collections can include millions of songs. However, time-on-task basically represents the time it takes users to peruse the review or summary, and the number of words in the summaries is significantly lower than the number of words in the reviews ($p < 0.001$, $t = 16.517$).

Hence, we also analyzed the influence of summaries on decision making, to see if they have any effects beyond cutting down on the number of words to read. Our assumption is that the default choice is to choose the first recommendation. Users in the baseline condition picked the first recommendation as often as the other two recommendations combined; users in the experimental condition picked the second and third recommendations more often than the first, and the difference between the two conditions is significant ($\chi^2 = 8.74$, $df = 1$, $p = 0.003$). If we examine behavior song by song, this holds true especially for song 3 ($\chi^2 = 12.3$, $df = 1$, $p < 0.001$) and song 4 ($\chi^2 = 5.08$, $df = 1$, $p = 0.024$). We speculate that users in the experimental condition

are more discriminatory in their choices, because important features of the recommended songs are evident in the summaries, but are buried in the album reviews. For example, for Song 3, only one of the 20 sentences in the album review is about the first recommended song, and is not very positive. Negative opinions are much more evident in the review summaries.

The questionnaires included three common questions between the two conditions. The experimental subjects gave a more positive assessment of the length of the summary than the baseline subjects ($p = 0.003$, $t = -3.248$, $df = 31.928$). There were no significant differences on the other two questions, feeling overwhelmed by the information provided; and whether the review/summary helped them to quickly make their selection.

A multiple Linear Regression with, as predictors, the number of words the user read before making the selection and the questions, and time on task as dependent variable, revealed only one, not surprising, correlation: the number of words the user read correlates with time on task ($R^2 = 0.277$, $\beta = 0.509$, $p = 0.004$).

Users in the experimental version were also asked to rate the grammaticality and coherence of the summary. The average rating was 3.33 for grammaticality, and 3.14 for coherence. Whereas these numbers in isolation are not too telling, they are at least suggestive that users did not find these summaries badly written. We found no significant correlations between grammaticality and coherence of summaries, and time on task.

6 Discussion and Conclusions

Most summarization research on customer reviews focuses on obtaining features of the products, but not much work has been done on presenting them as a coherent summary. In this paper, we described a system that uses information extraction and summarization techniques in order to generate summaries of individual songs from multiple album reviews. Whereas the techniques we have used are state-of-the-art, the contribution of our work is integrating them in an effective end-to-end system. We first evaluated it intrinsically as concerns information extraction, and the informativeness of the summaries. Perhaps more importantly, we also ran an extrinsic evaluation in the context of our prototype Music RS. Users made quicker decisions and their choice of recommendations was more varied when presented with song review summaries than with album reviews. Our framework can be applied to any domain where reviews of individual components need to be summarized from reviews of collections, such as travel reviews that cover many cities in a country, or different restaurants in a city.

References

Regina Barzilay and Kathleen McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Giuseppe Carenini, Raymond Ng, and Adam Pauls. 2006. Multi-document summarization of evaluative text. In *Proceedings of EACL*.

Oscar Clema. 2006. *Interaction Design for Recommender Systems*. Ph.D. thesis, Universitat Pompeu Fabra, Barcelona, July.

Marie-Catherine de Marnee and Christopher D. Manning. 2008. Stanford Typed Dependencies Manual. http://nlp.stanford.edu/software/dependencies_manual.pdf.

J. Stephen Downie and Xiao Hu. 2006. Review mining for music digital libraries: Phase ii. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 196–197, Chapel Hill, NC, USA.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC-06, the 5th Conference on Language Resources and Evaluation*, Genova, IT.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Michael Gamon, Anthony Aue, Simon Corston-Oliver, and Eric Ringger. 2005. Pulse: Mining customer opinions from free text. In *Advances in Intelligent Data Analysis VI*, volume 3646/2005 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin / Heidelberg.

Frederick Giasson and Yves Raimond. 2007. Music ontology specification. Working draft, February. http://pingthesemanticweb.com/ontology/mo/.

Aaron Harnly, Ani Nenkova, Rebecca Passonneau, and Owen Rambow. 2005. Automation of summary evaluation by the Pyramid method. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics*, Las Cruces, NM, June.

Ryuichiro Higashinaka, Rashmi Prasad, and Marilyn Walker. 2006. Learning to Generate Naturalistic Utterances Using Reviews in Spoken Dialogue Systems. In *COLING-ACL06*, Sidney, Australia.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD*, Seattle, Washington, USA, August.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10.

Susan McRoy, Songsak Ukul, and Syed Ali. 2003. An augmented template-based approach to text realization. In *Natural Language Engineering*, pages 381–420. Cambridge Press.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust, applied morphological generation. In *Proceedings of the 1st International Natural Language Generation Conference*.

Tom Mitchell. 1997. *Machine Learning*. McGraw Hill.

Vivi Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Patrick Nguyen, Milind Mahajan, and Geoffrey Zweig. 2007. Summarization of multiple user reviews in the restaurant domain. Technical Report MSR-TR-2007-126, Microsoft, September.

Markus Schedl, Gerhard Widmer, Tim Pohle, and Klaus Seyerlehner. 2007. Web-based detection of music band members and line-up. In *Proceedings of the Australian Computer Society*.

M. Soubbotin and S. Soubbotin. 2005. Trade-Off Between Factors Influencing Quality of the Summary. In *Document Understanding Workshop (DUC)*, Vancouver, BC, Canada.

Swati Tata. 2010. *SongRecommend: a Music Recommendation System with Fine-Grained Song Reviews*. Ph.D. thesis, University of Illinois, Chicago, IL.

2008. UPnP Device Architecture Version 1.0. (www.upnp.org).

Li Zhuang, Feng Jing, and Xiaoyan Zhu. 2006. Movie review mining and summarization. In *Conference on Information and Knowledge Management*, Arlington, Virginia, USA.

A study of Information Retrieval weighting schemes for sentiment analysis

Georgios Paltoglou
University of Wolverhampton
Wolverhampton, United Kingdom
g.paltoglou@wlv.ac.uk

Mike Thelwall
University of Wolverhampton
Wolverhampton, United Kingdom
m.thelwall@wlv.ac.uk

Abstract

Most sentiment analysis approaches use as baseline a support vector machines (SVM) classifier with binary unigram weights. In this paper, we explore whether more sophisticated feature weighting schemes from Information Retrieval can enhance classification accuracy. We show that variants of the classic $tf.idf$ scheme adapted to sentiment analysis provide significant increases in accuracy, especially when using a sublinear function for term frequency weights and document frequency smoothing. The techniques are tested on a wide selection of data sets and produce the best accuracy to our knowledge.

1 Introduction

The increase of user-generated content on the web in the form of reviews, blogs, social networks, tweets, fora, etc. has resulted in an environment where everyone can publicly express their opinion about events, products or people. This wealth of information is potentially of vital importance to institutions and companies, providing them with ways to research their consumers, manage their reputations and identify new opportunities. Wright (2009) claims that "for many businesses, online opinion has turned into a kind of virtual currency that can make or break a product in the marketplace".

Sentiment analysis, also known as opinion mining, provides mechanisms and techniques through which this vast amount of information can be processed and harnessed. Research in the field has mainly, but not exclusively, focused in two subproblems: detecting whether a segment of text, either a whole document or a sentence, is subjective or objective, i.e. contains an expression of opinion, and detecting the overall polarity of the text, i.e. positive or negative.

Most of the work in sentiment analysis has focused on supervised learning techniques (Sebastiani, 2002), although there are some notable exceptions (Turney, 2002; Lin and He, 2009). Previous research has shown that in general the performance of the former tend to be superior to that of the latter (Mullen and Collier, 2004; Lin and He, 2009). One of the main issues for supervised approaches has been the representation of documents. Usually a *bag of words* representation is adopted, according to which a document is modeled as an unordered collection of the words that it contains. Early research by Pang et al. (2002) in sentiment analysis showed that a binary unigram-based representation of documents, according to which a document is modeled only by the presence or absence of words, provides the best baseline classification accuracy in sentiment analysis in comparison to other more intricate representations using bigrams, adjectives, etc.

Later research has focused on extending the document representation with more complex features such as structural or syntactic information (Wilson et al., 2005), favorability measures from diverse sources (Mullen and Collier, 2004), implicit syntactic indicators (Greene and Resnik, 2009), stylistic and syntactic feature selection (Abbasi et al., 2008), "annotator rationales" (Zaidan et al., 2007) and others, but no systematic study has been presented exploring the benefits of employing more sophisticated models for assigning weights to word features.

In this paper, we examine whether term weighting functions adopted from Information Retrieval (IR) based on the standard $tf.idf$ formula and adapted to the particular setting of sentiment analysis can help classification accuracy. We demonstrate that variants of the original $tf.idf$ weighting scheme provide significant increases in classification performance. The advantages of the approach are that it is intuitive, computationally efficient

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1386–1395,
Uppsala, Sweden, 11-16 July 2010.

and doesn't require additional human annotation or external sources. Experiments conducted on a number of publicly available data sets improve on the previous state-of-the art.

The next section provides an overview of relevant work in sentiment analysis. In section 3 we provide a brief overview of the original $tf.idf$ weighting scheme along with a number of variants and show how they can be applied to a classification scenario. Section 4 describes the corpora that were used to test the proposed weighting schemes and section 5 discusses the results. Finally, we conclude and propose future work in section 6.

2 Prior Work

Sentiment analysis has been a popular research topic in recent years. Most of the work has focused on analyzing the content of movie or general product reviews, but there are also applications to other domains such as debates (Thomas et al., 2006; Lin et al., 2006), news (Devitt and Ahmad, 2007) and blogs (Ounis et al., 2008; Mishne, 2005). The book of Pang and Lee (2008) presents a thorough overview of the research in the field. This section presents the most relevant work.

Pang et al. (2002) conducted early polarity classification of reviews using supervised approaches. They employed Support Vector Machines (SVMs), Naive Bayes and Maximum Entropy classifiers using a diverse set of features, such as unigrams, bigrams, binary and term frequency feature weights and others. They concluded that sentiment classification is more difficult that standard topic-based classification and that using a SVM classifier with binary unigram-based features produces the best results.

A subsequent innovation was the detection and removal of the objective parts of documents and the application of a polarity classifier on the rest (Pang and Lee, 2004). This exploited text coherence with adjacent text spans which were assumed to belong to the same subjectivity or objectivity class. Documents were represented as graphs with sentences as nodes and association scores between them as edges. Two additional nodes represented the subjective and objective poles. The weights between the nodes were calculated using three different, heuristic decaying functions. Finding a partition that minimized a cost function separated the objective from the subjective sentences. They reported a statistically significant improvement over a Naive Bayes baseline using the whole text but only slight increase compared to using a SVM classifier on the entire document.

Mullen and Collier (2004) used SVMs and expanded the feature set for representing documents with favorability measures from a variety of diverse sources. They introduced features based on Osgood's Theory of Semantic Differentiation (Osgood, 1967) using WordNet to derive the values of potency, activity and evaluative of adjectives and Turney's semantic orientation (Turney, 2002). Their results showed that using a *hybrid* SVM classifier, that uses as features the distance of documents from the separating hyperplane, with all the above features produces the best results.

Whitelaw et al. (2005) added fine-grained semantic distinctions in the feature set. Their approach was based on a lexicon created in a semi-supervised fashion and then manually refined It consists of 1329 adjectives and their modifiers categorized under several taxonomies of appraisal attributes based on Martin and White's Appraisal Theory (2005). They combined the produced appraisal groups with unigram-based document representations as features to a Support Vector Machine classifier (Witten and Frank, 1999), resulting in significant increases in accuracy.

Zaidan et al. (2007) introduced "annotator rationales", i.e. words or phrases that explain the polarity of the document according to human annotators. By deleting rationale text spans from the original documents they created several *contrast* documents and constrained the SVM classifier to classify them less confidently than the originals. Using the largest training set size, their approach significantly increased the accuracy on a standard data set (see section 4).

Prabowo and Thelwall (2009) proposed a *hybrid* classification process by combining in sequence several ruled-based classifiers with a SVM classifier. The former were based on the General Inquirer lexicon (Wilson et al., 2005), the MontyLingua part-of-speech tagger (Liu, 2004) and co-occurrence statistics of words with a set of predefined reference words. Their experiments showed that combining multiple classifiers can result in better effectiveness than any individual classifier, especially when sufficient training data isn't available.

In contrast to machine learning approaches that require labeled corpora for training, Lin and

He (2009) proposed an unsupervised probabilistic modeling framework, based on Latent Dirichlet Allocation (LDA). The approach assumes that documents are a mixture of topics, i.e. probability distribution of words, according to which each document is generated through an hierarchical process and adds an extra sentiment layer to accommodate the opinionated nature (positive or negative) of the document. Their best attained performance, using a filtered subjectivity lexicon and removing objective sentences in a manner similar to Pang and Lee (2004), is only slightly lower than that of a fully-supervised approach.

3 A study of non-binary weights

We use the terms "features", "words" and "terms" interchangeably in this paper, since we mainly focus on unigrams. The approach nonetheless can easily be extended to higher order n-grams. Each document D therefore is represented as a bag-of-words feature vector: $D = \{w_1, w_2, ..., w_{|V|}\}$ where $|V|$ is the size of the vocabulary (i.e. the number of unique words) and w_i, $i = 1, \ldots, |V|$ is the weight of term i in document D.

Despite the significant attention that sentiment analysis has received in recent years, the best accuracy without using complex features (Mullen and Collier, 2004; Whitelaw et al., 2005) or additional human annotations (Zaidan et al., 2007) is achieved by employing a binary weighting scheme (Pang et al., 2002), where $w_i = 1, \text{if } tf_i > 0$ and $w_i = 0, \text{if } tf_i = 0$, where tf_i is the number of times that term i appears in document D (henceforth *raw term frequency*) and utilizing a SVM classifier. It is of particular interest that using tf_i in the document representation usually results in decreased accuracy, a result that appears to be in contrast with topic classification (Mccallum and Nigam, 1998; Pang et al., 2002).

In this paper, we also utilize SVMs but our study is centered on whether more sophisticated than binary or raw term frequency weighting functions can improve classification accuracy. We base our approach on the classic $tf.idf$ weighting scheme from Information Retrieval (IR) and adapt it to the domain of sentiment classification.

3.1 The classic tf.idf weighting schemes

The classic $tf.idf$ formula assigns weight w_i to term i in document D as:

$$w_i = tf_i \cdot idf_i = tf_i \cdot log\frac{N}{df_i} \quad (1)$$

where tf_i is the number of times term i occurs in D, idf_i is the *inverse document frequency* of term i, N is the total number of documents and df_i is the number of documents that contain term i.

The utilization of tf_i in classification is rather straightforward and intuitive but, as previously discussed, usually results in decreased accuracy in sentiment analysis. On the other hand, using idf to assign weights to features is less intuitive, since it only provides information about the general distribution of term i amongst documents of all classes, without providing any additional evidence of class preference. The utilization of idf in information retrieval is based on its ability to distinguish between content-bearing words (words with some semantical meaning) and simple function words, but this behavior is at least ambiguous in classification.

Table 1: SMART notation for *term frequency* variants. $max_t(tf)$ is the maximum frequency of any term in the document and avg_dl is the average number of terms in all the documents. For ease of reference, we also include the BM25 tf scheme. The k_1 and b parameters of BM25 are set to their default values of 1.2 and 0.95 respectively (Jones et al., 2000).

Notation	Term frequency
n (natural)	tf
l (logarithm)	$1 + log(tf)$
a (augmented)	$0.5 + \frac{0.5 \cdot tf}{max_t(tf)}$
b (boolean)	$\begin{cases} 1, & tf > 0 \\ 0, & otherwise \end{cases}$
L (log ave)	$\frac{1+log(tf)}{1+log(avg\_dl)}$
o (BM25)	$\frac{(k_1+1) \cdot tf}{k_1\left((1-b)+b \cdot \frac{dl}{avg\_dl}\right)+tf}$

3.2 Delta tf.idf

Martineau and Finin (2009) provide a solution to the above issue of idf utilization in a classification scenario by localizing the estimation of idf to the documents of one or the other class and subtracting the two values. Therefore, the weight of term

Table 2: SMART notation for *inverse document frequency* variants. For ease of reference we also include the BM25 idf factor and also present the extensions of the original formulations with their Δ variants.

Notation	Inverse Document Frequency
n (no)	1
t (idf)	$log\frac{N}{df}$
p (prob idf)	$log\frac{N-df}{df}$
k (BM25 idf)	$log\frac{N-df+0.5}{df+0.5}$
$\Delta(t)$ (Delta idf)	$log\frac{N_1 \cdot df_2}{N_2 \cdot df_1}$
$\Delta(t')$ (Delta smoothed idf)	$log\frac{N_1 \cdot df_2+0.5}{N_2 \cdot df_1+0.5}$
$\Delta(p)$ (Delta prob idf)	$log\frac{(N_1-df_1)\cdot df_2}{df_1 \cdot (N_2-df_2)}$
$\Delta(p')$ (Delta smoothed prob idf)	$log\frac{(N_1-df_1)\cdot df_2+0.5}{(N_2-df_2)\cdot df_1+0.5}$
$\Delta(k)$ (Delta BM25 idf)	$log\frac{(N_1-df_1+0.5)\cdot df_2+0.5}{(N_2-df_2+0.5)\cdot df_1+0.5}$

i in document D is estimated as:

$$\begin{aligned} w_i &= tf_i \cdot log_2(\frac{N_1}{df_{i,1}}) - tf_i \cdot log_2(\frac{N_2}{df_{i,2}}) \\ &= tf_i \cdot log_2(\frac{N_1 \cdot df_{i,2}}{df_{i,1} \cdot N_2}) \end{aligned} \quad (2)$$

where N_j is the total number of training documents in class c_j and $df_{i,j}$ is the number of training documents in class c_j that contain term i. The above weighting scheme was appropriately named $Delta\, tf.idf$.

The produced results (Martineau and Finin, 2009) show that the approach produces better results than the simple tf or binary weighting scheme. Nonetheless, the approach doesn't take into consideration a number of tested notions from IR, such as the non-linearity of term frequency to document relevancy (e.g. Robertson et al. (2004)) according to which, the probability of a document being relevant to a query term is typically sublinear in relation to the number of times a query term appears in the document. Additionally, their approach doesn't provide any sort of smoothing for the $df_{i,j}$ factor and is therefore susceptible to errors in corpora where a term occurs in documents of only one or the other class and therefore $df_{i,j} = 0$.

3.3 SMART and BM25 tf.idf variants

The SMART retrieval system by Salton (1971) is a retrieval system based on the vector space model (Salton and McGill, 1986). Salton and Buckley (1987) provide a number of variants of the $tf.idf$ weighting approach and present the *SMART notation scheme*, according to which each weighting function is defined by triples of letters; the first one denotes the term frequency factor, the second one corresponds to the inverse document frequency function and the last one declares the normalization that is being applied. The upper rows of tables 1, 2 and 3 present the three most commonly used weighting functions for each factor respectively. For example, a binary document representation would be equivalent to $SMART.bnn$[1] or more simply bnn, while a simple raw term frequency based would be notated as nnn or nnc with cosine normalization.

Table 3: SMART normalization.

Notation	Normalization
n (none)	1
c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+...+w_n^2}}$

Significant research has been done in IR on diverse weighting functions and not all versions of SMART notations are consistent (Manning et al., 2008). Zobel and Moffat (1998) provide an exhaustive study but in this paper, due to space constraints, we will follow the concise notation presented by Singhal et al. (1995).

The BM25 weighting scheme (Robertson et al., 1994; Robertson et al., 1996) is a probabilistic model for information retrieval and is one of the most popular and effective algorithms used in information retrieval. For ease of reference, we incorporate the BM25 tf and idf factors into the SMART annotation scheme (last row of table 1 and 4^{th} row of table 2), therefore the weight w_i of term i in document D according to the BM25 scheme is notated as $SMART.okn$ or okn.

Most of the tf weighting functions in SMART and the BM25 model take into consideration the non-linearity of document relevance to term fre-

[1]Typically, a weighting function in the SMART system is defined as a pair of triples, i.e. $ddd.qqq$ where the first triple corresponds to the document representation and the second to the query representation. In the context that the SMART annotation is used here, we will use the prefix $SMART$ for the first part and a triple for the document representation in the second part, i.e. $SMART.ddd$, or more simply ddd.

quency and thus employ tf factors that scale sublinearly in relation to term frequency. Additionally, the BM25 tf variant also incorporates a scaling for the length of the document, taking into consideration that longer documents will by definition have more term occurences[2]. Effective weighting functions is a very active research area in information retrieval and it is outside the scope of this paper to provide an in-depth analysis but significant research can be found in Salton and McGill (1986), Robertson et al. (2004), Manning et al. (2008) or Armstrong et al. (2009) for a more recent study.

3.4 Introducing SMART and BM25 Delta tf.idf variants

We apply the idea of localizing the estimation of idf values to documents of one class but employ more sophisticated term weighting functions adapted from the SMART retrieval system and the BM25 probabilistic model. The resulting idf weighting functions are presented in the lower part of table 2. We extend the original SMART annotation scheme by adding Delta (Δ) variants of the original idf functions and additionally introduce smoothed Delta variants of the *idf* and the *prob idf* factors for completeness and comparative reasons, noted by their *accented* counterparts. For example, the weight of term i in document D according to the $o\Delta(k)n$ weighting scheme where we employ the BM25 tf weighting function and utilize the difference of class-based BM25 idf values would be calculated as:

$$\begin{aligned} w_i &= \frac{(k_1+1)\cdot tf_i}{K+tf_i}\cdot log(\frac{N_1-df_{i,1}+0.5}{df_{i,1}+0.5}) \\ &- \frac{(k_1+1)\cdot tf_i}{K+tf_i}\cdot log(\frac{N_2-df_{i,2}+0.5}{df_{i,2}+0.5}) \\ &= \frac{(k_1+1)\cdot tf_i}{K+tf_i} \\ &\cdot \quad log\left(\frac{(N_1-df_{i,1}+0.5)\cdot(df_{i,2}+0.5)}{(N_2-df_{i,2}+0.5)\cdot(df_{i,1}+0.5)}\right) \end{aligned}$$

where K is defined as $k_1\left((1-b)+b\cdot\frac{dl}{avg\_dl}\right)$. However, we used a minor variation of the above formulation for all the final *accented* weighting functions in which the smoothing factor is added to the product of df_i with N_i (or its variation for $\Delta(p')$ and $\Delta(k)$), rather than to the df_i alone as the above formulation would imply (see table 2). The above variation was made for two reasons: firstly, when the df_i's are larger than 1 then the smoothing factor influences the final idf value only in a minor way in the revised formulation, since it is added only after the multiplication of the df_i with N_i (or its variation). Secondly, when $df_i = 0$, then the smoothing factor correctly adds only a small mass, avoiding a potential division by zero, where otherwise it would add a much greater mass, because it would be multiplied by N_i.

According to this annotation scheme therefore, the original approach by Martineau and Finin (2009) can be represented as $n\Delta(t)n$.

We hypothesize that the utilization of sophisticated term weighting functions that have proved effective in information retrieval, thus providing an indication that they appropriately model the distinctive power of terms to documents and the smoothed, localized estimation of idf values will prove beneficial in sentiment classification.

Table 4: Reported accuracies on the Movie Review data set. Only the best reported accuracy for each approach is presented, measured by 10-fold cross validation. The list is not exhaustive and because of differences in training/testing data splits the results are not directly comparable. It is produced here only for reference.

Approach	Acc.
SVM with unigrams & binary weights (Pang et al., 2002), reported at (Pang and Lee, 2004)	87.15%
Hybrid SVM with Turney/Osgood Lemmas (Mullen and Collier, 2004)	86%
SVM with min-cuts (Pang and Lee, 2004)	87.2%
SVM with appraisal groups (Whitelaw et al., 2005)	90.2%
SVM with log likehood ratio feature selection (Aue and Gamon, 2005)	90.45%
SVM with annotator rationales (Zaidan et al., 2007)	92.2%
LDA with filtered lexicon, subjectivity detection (Lin and He, 2009)	84.6%

The approach is straightforward, intuitive, computationally efficient, doesn't require additional human effort and takes into consideration standardized and tested notions from IR. The results presented in section 5 show that a number

[2]We deliberately didn't extract the normalization component from the BM25 tf variant, as that would unnecessarily complicate the notation.

of weighting functions solidly outperform other state-of-the-art approaches. In the next section, we present the corpora that were used to study the effectiveness of different weighting schemes.

4 Experimental setup

We have experimented with a number of publicly available data sets.

The movie review dataset by Pang et al. (2002) has been used extensively in the past by a number of researchers (see Table 4), presenting the opportunity to compare the produced results with previous approaches. The dataset comprises 2,000 movie reviews, equally divided between positive and negative, extracted from the Internet Movie Database[3] archive of the *rec.arts.movies.reviews* newsgroup. In order to avoid reviewer bias, only 20 reviews per author were kept, resulting in a total of 312 reviewers[4]. The best attained accuracies by previous research on the specific data are presented in table 4. We do not claim that those results are directly comparable to ours, because of potential subtle differences in tokenization, classifier implementations etc, but we present them here for reference.

The Multi-Domain Sentiment data set (MDSD) by Blitzer et al. (2007) contains Amazon reviews for four different product types: books, electronics, DVDs and kitchen appliances. Reviews with ratings of 3 or higher, on a 5-scale system, were labeled as positive and reviews with a rating less than 3 as negative. The data set contains 1,000 positive and 1,000 negative reviews for each product category for a total of 8,000 reviews. Typically, the data set is used for domain adaptation applications but in our setting we only split the reviews between positive and negative[5].

Lastly, we present results from the BLOGS06 (Macdonald and Ounis, 2006) collection that is comprised of an uncompressed 148GB crawl of approximately 100,000 blogs and their respective RSS feeds. The collection has been used for 3 consecutive years by the Text REtrieval Conferences (TREC)[6]. Participants of the conference are provided with the task of finding documents (i.e. web pages) expressing an opinion about specific entities X, which may be people, companies, films etc. The results are given to human assessors who then judge the content of the webpages (i.e. blog post and comments) and assign each webpage a score: "1" if the document contains relevant, factual information about the entity but no expression of opinion, "2" if the document contains an explicit negative opinion towards the entity and "4" is the document contains an explicit positive opinion towards the entity. We used the produced assessments from all 3 years of the conference in our data set, resulting in 150 different entity searches and, after duplicate removal, 7,930 negative documents (i.e. having an assessment of "2") and 9,968 positive documents (i.e. having an assessment of "4"), which were used as the "gold standard" [7]. Documents are annotated at the document-level, rather than at the post level, making this data set somewhat noisy. Additionally, the data set is particularly large compared to the other ones, making classification especially challenging and interesting. More information about all data sets can be found at table 5.

We have kept the pre-processing of the documents to a minimum. Thus, we have lower-cased all words and removed all punctuation but we have not removed stop words or applied stemming. We have also refrained from removing words with low or high occurrence. Additionally, for the BLOGS06 data set, we have removed all html formatting.

We utilize the implementation of a support vector classifier from the *LIBLINEAR* library (Fan et al., 2008). We use a linear kernel and default parameters. All results are based on leave-one out cross validation accuracy. The reason for this choice of cross-validation setting, instead of the most standard ten-fold, is that all of the proposed approaches that use some form of idf utilize the training documents for extracting document frequency statistics, therefore more information is available to them in this experimental setting.

Because of the high number of possible combinations between tf and idf variants ($6 \cdot 9 \cdot 2 = 108$) and due to space constraints we only present results from a subset of the most representative combinations. Generally, we'll use the cosine normalized variants of unsmoothed delta weighting schemes, since they perform better than their un-

[3] http://www.imdb.com

[4] The dataset can be found at: http://www.cs.cornell.edu/People/pabo/movie-review-data/review_polarity.tar.gz.

[5] The data set can be found at http://www.cs.jhu.edu/mdredze/datasets/sentiment/

[6] http://www.trec.nist.gov

[7] More information about the data set, as well as information on how it can be obtained can be found at: http://ir.dcs.gla.ac.uk/test_collections/blogs06info.html

Table 5: Statistics about the data sets used.

Data set	#Documents	#Terms	#Unique Terms	Average #Terms per Document
Movie Reviews	2,000	1,336,883	39,399	668
Multi-Domain Sentiment Dataset (MDSD)	8,000	1,741,085	455,943	217
BLOGS06	17,898	51,252,850	367,899	2,832

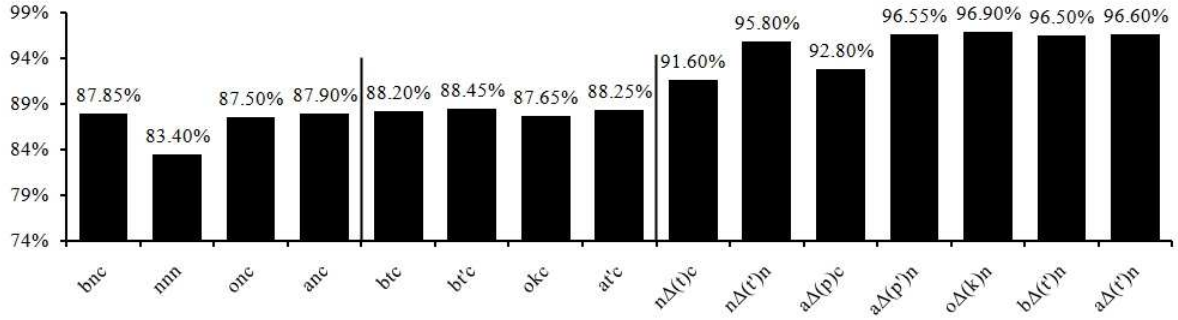


Figure 1: Reported accuracy on the Movie Review data set.

normalized counterparts. We'll avoid using normalization for the smoothed versions, in order to focus our attention on the results of smoothing, rather than normalization.

5 Results

Results for the Movie Reviews, Multi-Domain Sentiment Dataset and BLOGS06 corpora are reported in figures 1, 2 and 3 respectively.

On the Movie Review data set, the results reconfirm that using binary features (bnc) is better than raw term frequency (nnc) (83.40%) features. For reference, in this setting the unnormalized vector using the raw tf approach (nnn) performs similar to the normalized (nnc) (83.40% vs. 83.60%), the former not present in the graph. Nonetheless, using any scaled tf weighting function (anc or onc) performs as well as the binary approach (87.90% and 87.50% respectively). Of interest is the fact that although the BM25 tf algorithm has proved much more successful in IR, the same doesn't apply in this setting and its accuracy is similar to the simpler *augmented tf* approach.

Incorporating un-localized variants of idf (middle graph section) produces only small increases in accuracy. Smoothing also doesn't provide any particular advantage, e.g. btc (88.20%) vs. $bt'c$ (88.45%), since no zero idf values are present. Again, using more sophisticated tf functions provides an advantage over raw tf, e.g. $nt'c$ attains an accuracy of 86.6% in comparison to $at'c$'s 88.25%, although the simpler $at'c$ is again as effective than the BM25 tf ($ot'c$), which performs at 88%. The actual idf weighting function is of some importance, e.g. $ot'c$ (88%) vs. okc (87.65%) and akc (88%) vs. $at'c$ (88.25%), with simpler idf factors performing similarly, although slightly better than BM25.

Introducing smoothed, localized variants of idf and scaled or binary tf weighting schemes produces significant advantages. In this setting, smoothing plays a role, e.g. $n\Delta(t)c$[8] (91.60%) vs. $n\Delta(t')n$ (95.80%) and $a\Delta(p)c$ (92.80%) vs. $a\Delta(p')n$ (96.55%), since we can expect zero class-based estimations of idf values, supporting our initial hypothesis on its importance. Additionally, using *augmented*, BM25 or binary tf weights is always better than raw term frequency, providing further support on the advantages of using sublinear tf weighting functions[9]. In this setting, the best accuracy of 96.90% is attained using BM25 tf weights with the BM25 delta idf variant, although binary or *augmented* tf weights using

[8]The original *Delta tf.idf* by Martineau and Finin (2009) has a limitation of utilizing features with $df > 2$. In our experiments it performed similarly to $n\Delta(t)n$ (90.60%) but still lower than the $cosine$ normalized variant $n\Delta(t)c$ included in the graph (91.60%).

[9]Although not present in the graph, for completeness reasons it should be noted that $l\Delta(s)n$ and $L\Delta(s)n$ also perform very well, both reaching accuracies of approx. 96%.

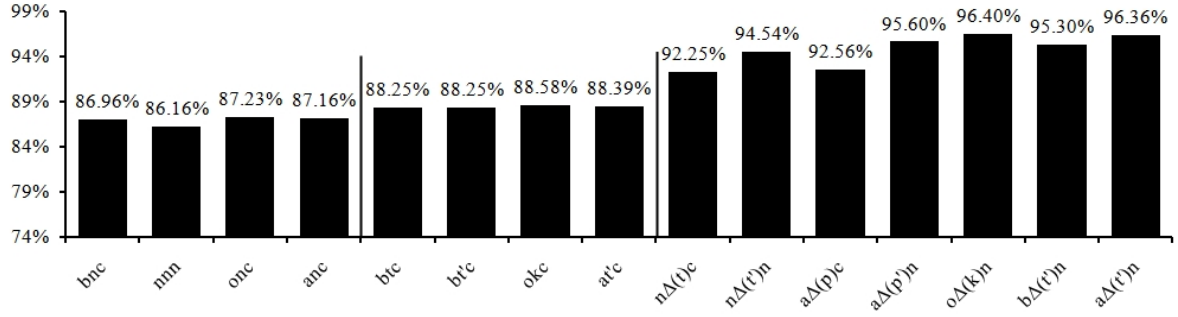


Figure 2: Reported accuracy on the Multi-Domain Sentiment data set.

delta idf perform similarly (96.50% and 96.60% respectively). The results indicate that the tf and the idf factor themselves aren't of significant importance, as long as the former are scaled and the latter smoothed in some manner. For example, $a\Delta(p')n$ vs. $a\Delta(t')n$ perform quite similarly.

The results from the Multi-Domain Sentiment data set (figure 2) largely agree with the findings on the Movie Review data set, providing a strong indication that the approach isn't limited to a specific domain. Binary weights outperform *raw term frequency* weights and perform similarly with scaled tf's. Non-localized variants of idf weights do provide a small advantage in this data set although the actual idf variant isn't important, e.g. btc, $bt'c$, and okc all perform similarly. The utilized tf variant also isn't important, e.g. $at'c$ (88.39%) vs. $bt'c$ (88.25%).

We focus our attention on the $delta\ idf$ variants which provide the more interesting results. The importance of smoothing becomes apparent when comparing the accuracy of $a\Delta(p)c$ and its smoothed variant $a\Delta(p')n$ (92.56% vs. 95.6%). Apart from that, all smoothed $delta\ idf$ variants perform very well in this data set, including somewhat surprisingly, $n\Delta(t')n$ which uses raw tf (94.54%). Considering that the average tf per document is approx. 1.9 in the Movie Review data set and 1.1 in the MDSD, the results can be attributed to the fact that words tend to typically appear only once per document in the latter, therefore minimizing the difference of the weights attributed by different tf functions[10]. The best attained accuracy is 96.40% but as the MDSD has mainly been used for domain adaptation applications, there is no clear baseline to compare it with.

[10]For reference, the average tf per document in the BLOGS06 data set is 2.4.

Lastly, we present results on the BLOGS06 dataset in figure 3. As previously noted, this data set is particularly noisy, because it has been annotated at the document-level rather than the post-level and as a result, the differences aren't as profound as in the previous corpora, although they do follow the same patterns. Focusing on the delta idf variants, the importance of smoothing becomes apparent, e.g. $a\Delta(p)c$ vs. $a\Delta(p')n$ and $n\Delta(t)c$ vs. $n\Delta(t')n$. Additionally, because of the fact that documents tend to be more verbose in this data set, the scaled tf variants also perform better than the simple *raw tf* ones, $n\Delta(t')n$ vs. $a\Delta(t')n$. Lastly, as previously, the smoothed localized idf variants perform better than their unsmoothed counterparts, e.g. $n\Delta(t)n$ vs. $n\Delta(t')n$ and $a\Delta(p)c$ vs. $a\Delta(p')n$.

6 Conclusions

In this paper, we presented a study of document representations for sentiment analysis using term weighting functions adopted from information retrieval and adapted to classification. The proposed weighting schemes were tested on a number of publicly available datasets and a number of them repeatedly demonstrated significant increases in accuracy compared to other state-of-the-art approaches. We demonstrated that for accurate classification it is important to use term weighting functions that scale sublinearly in relation to the number of times a term occurs in a document and that document frequency smoothing is a significant factor.

In the future we plan to test the proposed weighting functions in other domains such as topic classification and additionally extend the approach to accommodate multi-class classification.

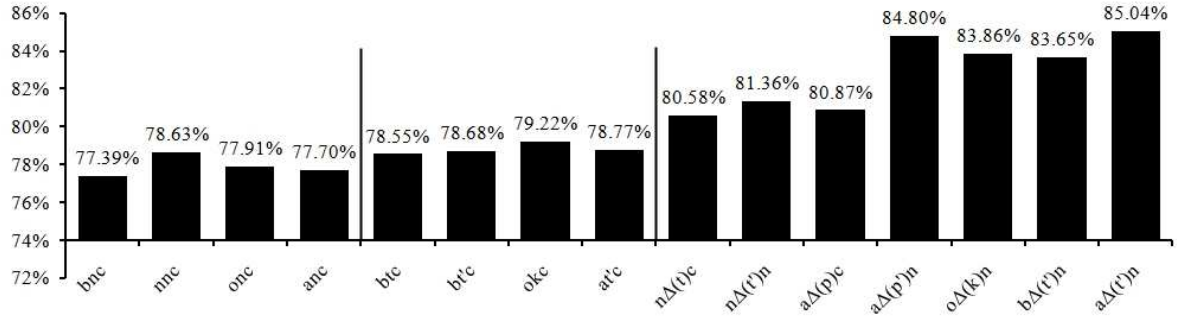


Figure 3: Reported accuracy on the BLOGS06 data set.

Acknowledgments

This work was supported by a European Union grant by the 7th Framework Programme, Theme 3: Science of complex systems for socially intelligent ICT. It is part of the CyberEmotions Project (Contract 231323).

References

Ahmed Abbasi, Hsinchun Chen, and Arab Salem. 2008. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Trans. Inf. Syst.*, 26(3):1–34.

Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements that don't add up: ad-hoc retrieval results since 1998. In David Wai Lok Cheung, Il Y. Song, Wesley W. Chu, Xiaohua Hu, Jimmy J. Lin, David Wai Lok Cheung, Il Y. Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *CIKM*, pages 601–610, New York, NY, USA. ACM.

Anthony Aue and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 984–991, Prague, Czech Republic, June. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 503–511, Boulder, Colorado, June. Association for Computational Linguistics.

K. Sparck Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 375–384, New York, NY, USA. ACM.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Hugo Liu. 2004. MontyLingua: An end-to-end natural language processor with common sense. Technical report, MIT.

C. Macdonald and I. Ounis. 2006. The trec blogs06 collection : Creating and analysing a blog test collection. *DCS Technical Report Series*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July.

J. R. Martin and P. R. R. White. 2005. *The language of evaluation : appraisal in English / J.R. Martin and P.R.R. White*. Palgrave Macmillan, Basingstoke :.

Justin Martineau and Tim Finin. 2009. Delta TFIDF: An Improved Feature Space for Sentiment Analysis. In *Proceedings of the Third AAAI Internatonal Conference on Weblogs and Social Media*, San Jose, CA, May. AAAI Press. (poster paper).

A. Mccallum and K. Nigam. 1998. A comparison of event models for naive bayes text classification.

G. Mishne. 2005. Experiments with mood classification in blog posts. In *1st Workshop on Stylistic Analysis Of Text For Information Access*.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 412–418, Barcelona, Spain, July. Association for Computational Linguistics.

Charles E. Osgood. 1967. *The measurement of meaning / [by] [Charles E. Osgood, George J. Suci [and] Percy H. Tannenbaum]*. University of Illinois Press, Urbana :, 2nd ed. edition.

Iadh Ounis, Craig Macdonald, and Ian Soboroff. 2008. Overview of the trec-2008 blog trac. In *The Seventeenth Text REtrieval Conference (TREC 2008) Proceedings*. NIST.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *In Proceedings of the ACL*, pages 271–278.

B. Pang and L. Lee. 2008. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc.

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

Rudy Prabowo and Mike Thelwall. 2009. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, April.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *TREC*, pages 0–.

S E Robertson, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. 1996. Okapi at trec-2. In *In The Second Text REtrieval Conference (TREC-2), NIST Special Special Publication 500-215*, pages 21–34.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA. ACM.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

G. Salton. 1971. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1ñ47.

Amit Singhal, Gerard Salton, and Chris Buckley. 1995. Length normalization in degraded text collections. Technical report, Ithaca, NY, USA.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. *CoRR*, abs/cs/0607062.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. 2005. Using appraisal groups for sentiment analysis. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631, New York, NY, USA. ACM.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, CA.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 1st edition, October.

Alex Wright. 2009. Mining the web for feelings, not facts. August 23, NY Times, last accessed October 2, 2009, http://http://www.nytimes.com/2009/08/24/technology/internet/ 24emotion.html?_r=1.

O.F. Zaidan, J. Eisner, and C.D. Piatko. 2007. Using Annotator Rationales to Improve Machine Learning for Text Categorization. *Proceedings of NAACL HLT*, pages 260–267.

Justin Zobel and Alistair Moffat. 1998. Exploring the similarity space. *SIGIR Forum*, 32(1):18–34.

Supervised Noun Phrase Coreference Research: The First Fifteen Years

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

The research focus of computational coreference resolution has exhibited a shift from heuristic approaches to machine learning approaches in the past decade. This paper surveys the major milestones in supervised coreference research since its inception fifteen years ago.

1 Introduction

Noun phrase (NP) coreference resolution, the task of determining which NPs in a text or dialogue refer to the same real-world entity, has been at the core of natural language processing (NLP) since the 1960s. NP coreference is related to the task of anaphora resolution, whose goal is to identify an antecedent for an *anaphoric* NP (i.e., an NP that depends on another NP, specifically its antecedent, for its interpretation) [see van Deemter and Kibble (2000) for a detailed discussion of the difference between the two tasks]. Despite its simple task definition, coreference is generally considered a difficult NLP task, typically involving the use of sophisticated knowledge sources and inference procedures (Charniak, 1972). Computational theories of discourse, in particular *focusing* (see Grosz (1977) and Sidner (1979)) and *centering* (Grosz et al. (1983; 1995)), have heavily influenced coreference research in the 1970s and 1980s, leading to the development of numerous *centering algorithms* (see Walker et al. (1998)).

The focus of coreference research underwent a gradual shift from heuristic approaches to machine learning approaches in the 1990s. This shift can be attributed in part to the advent of the statistical NLP era, and in part to the public availability of annotated coreference corpora produced as part of the MUC-6 (1995) and MUC-7 (1998) conferences. Learning-based coreference research has remained vibrant since then, with results regularly published not only in general NLP conferences, but also in specialized conferences (e.g., the biennial Discourse Anaphora and Anaphor Resolution Colloquium (DAARC)) and workshops (e.g., the series of Bergen Workshop on Anaphora Resolution (WAR)). Being inherently a *clustering* task, coreference has also received a lot of attention in the machine learning community.

Fifteen years have passed since the first paper on learning-based coreference resolution was published (Connolly et al., 1994). Our goal in this paper is to provide NLP researchers with a survey of the major milestones in *supervised* coreference research, focusing on the computational models, the linguistic features, the annotated corpora, and the evaluation metrics that were developed in the past fifteen years. Note that several leading coreference researchers have published books (e.g., Mitkov (2002)), written survey articles (e.g., Mitkov (1999), Strube (2009)), and delivered tutorials (e.g., Strube (2002), Ponzetto and Poesio (2009)) that provide a broad overview of coreference research. This survey paper aims to *complement*, rather than supersede, these previously published materials. In particular, while existing survey papers discuss learning-based coreference research primarily in the context of the influential mention-pair model, we additionally survey recently proposed learning-based coreference models, which attempt to address the weaknesses of the mention-pair model. Due to space limitations, however, we will restrict our discussion to the most commonly investigated kind of coreference relation: the *identity* relation for NPs, excluding coreference among clauses and bridging references (e.g., part/whole and set/subset relations).

2 Annotated Corpora

The widespread popularity of machine learning approaches to coreference resolution can be attributed in part to the public availability of an-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1396–1411,
Uppsala, Sweden, 11-16 July 2010.

notated coreference corpora. The MUC-6 and MUC-7 corpora, though relatively small (60 documents each) and homogeneous w.r.t. document type (newswire articles only), have been extensively used for training and evaluating coreference models. Equally popular are the corpora produced by the Automatic Content Extraction (ACE[1]) evaluations in the past decade: while the earlier ACE corpora (e.g., ACE-2) consist of solely English newswire and broadcast news articles, the later ones (e.g., ACE 2005) have also included Chinese and Arabic documents taken from additional sources such as broadcast conversations, webblog, usenet, and conversational telephone speech.

Coreference annotations are also publicly available in treebanks. These include (1) the English Penn Treebank (Marcus et al., 1993), which is labeled with coreference links as part of the OntoNotes project (Hovy et al., 2006); (2) the Tübingen Treebank (Telljohann et al., 2004), which is a collection of German news articles consisting of 27,125 sentences; (3) the Prague Dependency Treebank (Hajič et al., 2006), which consists of 3168 news articles taken from the Czech National Corpus; (4) the NAIST Text Corpus (Iida et al., 2007b), which consists of 287 Japanese news articles; (5) the AnCora Corpus (Recasens and Martí, 2009), which consists of Spanish and Catalan journalist texts; and (6) the GENIA corpus (Ohta et al., 2002), which contains 2000 MEDLINE abstracts.

Other publicly available coreference corpora of interest include two annotated by Ruslan Mitkov's research group: (1) a 55,000-word corpus in the domain of security/terrorism (Hasler et al., 2006); and (2) training data released as part of the 2007 Anaphora Resolution Exercise (Orăsan et al., 2008), a coreference resolution shared task. There are also two that consist of spoken dialogues: the TRAINS93 corpus (Heeman and Allen, 1995) and the Switchboard data set (Calhoun et al., in press).

Additional coreference data will be available in the near future. For instance, the SemEval-2010 shared task on Coreference Resolution in Multiple Languages (Recasens et al., 2009) has promised to release coreference data in six languages. In addition, Massimo Poesio and his colleagues are leading an annotation project that aims to collect large amounts of coreference data for English via a Web Collaboration game called Phrase Detectives[2].

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

[2] http://www.phrasedetectives.org

3 Learning-Based Coreference Models

In this section, we examine three important classes of coreference models that were developed in the past fifteen years, namely, the mention-pair model, the entity-mention model, and ranking models.

3.1 Mention-Pair Model

The mention-pair model is a classifier that determines whether two NPs are coreferent. It was first proposed by Aone and Bennett (1995) and McCarthy and Lehnert (1995), and is one of the most influential learning-based coreference models. Despite its popularity, this binary classification approach to coreference is somewhat undesirable: the transitivity property inherent in the coreference relation cannot be enforced, as it is possible for the model to determine that A and B are coreferent, B and C are coreferent, but A and C are not coreferent. Hence, a separate clustering mechanism is needed to coordinate the pairwise classification decisions made by the model and construct a coreference partition.

Another issue that surrounds the acquisition of the mention-pair model concerns the way training instances are created. Specifically, to determine whether a pair of NPs is coreferent or not, the mention-pair model needs to be trained on a data set where each instance represents two NPs and possesses a class value that indicates whether the two NPs are coreferent. Hence, a natural way to assemble a training set is to create one instance from each pair of NPs appearing in a training document. However, this instance creation method is rarely employed: as most NP pairs in a text are not coreferent, this method yields a training set with a skewed class distribution, where the negative instances significantly outnumber the positives.

As a result, in practical implementations of the mention-pair model, one needs to specify not only the *learning algorithm* for training the model and the *linguistic features* for representing an instance, but also the *training instance creation method* for reducing class skewness and the *clustering algorithm* for constructing a coreference partition.

3.1.1 Creating Training Instances

As noted above, the primary purpose of training instance creation is to reduce class skewness. Many heuristic instance creation methods have been proposed, among which Soon et al.'s (1999; 2001) is arguably the most popular choice. Given

an anaphoric noun phrase[3], NP_k, Soon et al.'s method creates a *positive instance* between NP_k and its closest preceding antecedent, NP_j, and a *negative instance* by pairing NP_k with each of the intervening NPs, $\text{NP}_{j+1}, \ldots, \text{NP}_{k-1}$.

With an eye towards improving the precision of a coreference resolver, Ng and Cardie (2002c) propose an instance creation method that involves a single modification to Soon et al.'s method: if NP_k is non-pronominal, a positive instance should be formed between NP_k and its closest preceding *non-pronominal* antecedent instead. This modification is motivated by the observation that it is not easy for a human, let alone a machine learner, to learn from a positive instance where the antecedent of a non-pronominal NP is a pronoun.

To further reduce class skewness, some researchers employ a filtering mechanism on top of an instance creation method, thereby disallowing the creation of training instances from NP pairs that are unlikely to be coreferent, such as NP pairs that violate gender and number agreement (e.g., Strube et al. (2002), Yang et al. (2003)).

While many instance creation methods are heuristic in nature (see Uryupina (2004) and Hoste and Daelemans (2005)), some are learning-based. For example, motivated by the fact that some coreference relations are harder to identify than the others (see Harabagiu et al. (2001)), Ng and Cardie (2002a) present a method for mining easy positive instances, in an attempt to avoid the inclusion of hard training instances that may complicate the acquisition of an accurate coreference model.

3.1.2 Training a Coreference Classifier

Once a training set is created, we can train a coreference model using an off-the-shelf learning algorithm. Decision tree induction systems (e.g., C5 (Quinlan, 1993)) are the first and one of the most widely used learning algorithms by coreference researchers, although rule learners (e.g., RIPPER (Cohen, 1995)) and memory-based learners (e.g., TiMBL (Daelemans and Van den Bosch, 2005)) are also popular choices, especially in early applications of machine learning to coreference resolution. In recent years, statistical learners such as maximum entropy models (Berger et al., 1996), voted perceptrons (Freund and Schapire, 1999), and support vector machines (Joachims, 1999) have been increasingly used, in part due to their ability to provide a confidence value (e.g., in the form of a probability) associated with a classification, and in part due to the fact that they can be easily adapted to train recently proposed ranking-based coreference models (see Section 3.3).

3.1.3 Generating an NP Partition

After training, we can apply the resulting model to a test text, using a clustering algorithm to coordinate the pairwise classification decisions and impose an NP partition. Below we describe some commonly used coreference clustering algorithms.

Despite their simplicity, *closest-first clustering* (Soon et al., 2001) and *best-first clustering* (Ng and Cardie, 2002c) are arguably the most widely used coreference clustering algorithms. The closest-first clustering algorithm selects as the antecedent for an NP, NP_k, the closest preceding noun phrase that is classified as coreferent with it.[4] However, if no such preceding noun phrase exists, no antecedent is selected for NP_k. The *best-first* clustering algorithm aims to improve the precision of closest-first clustering, specifically by selecting as the antecedent of NP_k the *most probable* preceding NP that is classified as coreferent with it.

One criticism of the closest-first and best-first clustering algorithms is that they are too greedy. In particular, clusters are formed based on a small subset of the pairwise decisions made by the model. Moreover, positive pairwise decisions are unjustifiably favored over their negative counterparts. For example, three NPs are likely to end up in the same cluster in the resulting partition even if there is strong evidence that A and C are not coreferent, as long as the other two pairs (i.e., (A,B) and (B,C)) are classified as positive.

Several algorithms that address one or both of these problems have been used for coreference clustering. *Correlation clustering* (Bansal et al., 2002), which produces a partition that respects as many pairwise decisions as possible, is used by McCallum and Wellner (2004), Zelenko et al. (2004), and Finley and Joachims (2005). *Graph partitioning algorithms* are applied on a weighted, undirected graph where a vertex corresponds to an NP and an edge is weighted by the pairwise coreference scores between two NPs (e.g., McCallum and Wellner (2004), Nicolae and Nico-

[3] In this paper, we use the term *anaphoric* to describe any NP that is part of a coreference chain but is not the head of the chain. Hence, proper names can be anaphoric under this overloaded definition, but linguistically, they are not.

[4] If a probabilistic model is used, we can define a threshold above which a pair of NPs is considered coreferent.

lae (2006)). The *Dempster-Shafer rule* (Dempster, 1968), which combines the positive and negative pairwise decisions to score a partition, is used by Kehler (1997) and Bean and Riloff (2004) to identify the most probable NP partition.

Some clustering algorithms bear a closer resemblance to the way a human creates coreference clusters. In these algorithms, not only are the NPs in a text processed in a left-to-right manner, the later coreference decisions are dependent on the earlier ones (Cardie and Wagstaff, 1999; Klenner and Ailloud, 2008).[5] For example, to resolve an NP, NP_k, Cardie and Wagstaff's algorithm considers each preceding NP, NP_j, as a candidate antecedent in a right-to-left order. If NP_k and NP_j are likely to be coreferent, the algorithm imposes an additional check that NP_k does not violate any constraint on coreference (e.g., gender agreement) with any NP in the cluster containing NP_j before positing that the two NPs are coreferent.

Luo et al.'s (2004) Bell-tree-based algorithm is another clustering algorithm where the later coreference decisions are dependent on the earlier ones. A Bell tree provides an elegant way of organizing the space of NP partitions. Informally, a node in the ith level of a Bell tree corresponds to an ith-order *partial* partition (i.e., a partition of the first i NPs of the given document), and the ith level of the tree contains *all* possible ith-order partial partitions. Hence, a leaf node contains a *complete* partition of the NPs, and the goal is to search for the leaf node that contains the most probable partition. The search starts at the root, and a partitioning of the NPs is incrementally constructed as we move down the tree. Specifically, based on the coreference decisions it has made in the first $i-1$ levels of the tree, the algorithm determines at the ith level whether the ith NP should start a new cluster, or to which preceding *cluster* it should be assigned.

While many coreference clustering algorithms have been developed, there have only been a few attempts to compare their effectiveness. For example, Ng and Cardie (2002c) report that best-first clustering is better than closest-first clustering. Nicolae and Nicolae (2006) show that best-first clustering performs similarly to Bell-tree-based clustering, but neither of these algorithms performs as well as their proposed minimum-cut-based graph partitioning algorithm.

[5] When applying closest-first and best-first clustering, Soon et al. (2001) and Ng and Cardie (2002c) also process the NPs in a sequential manner, but since the later decisions are not dependent on the earlier ones, the order in which the NPs are processed does not affect their clustering results.

3.1.4 Determining NP Anaphoricity

While coreference clustering algorithms attempt to resolve *each* NP encountered in a document, only a subset of the NPs are *anaphoric* and therefore need to be resolved. Hence, knowledge of the anaphoricity of an NP can potentially improve the precision of a coreference resolver.

Traditionally, the task of anaphoricity determination has been tackled independently of coreference resolution using a variety of techniques. For example, pleonastic *it* has been identified using heuristic approaches (e.g., Paice and Husk (1987), Lappin and Leass (1994), Kennedy and Boguraev (1996)), supervised approaches (e.g., Evans (2001), Müller (2006), Versley et al. (2008a)), and distributional methods (e.g., Bergsma et al. (2008)); and non-anaphoric definite descriptions have been identified using rule-based techniques (e.g., Vieira and Poesio (2000)) and unsupervised techniques (e.g., Bean and Riloff (1999)).

Recently, anaphoricity determination has been evaluated in the context of coreference resolution, with results showing that training an anaphoricity classifier to identify and filter non-anaphoric NPs prior to coreference resolution can improve a learning-based resolver (e.g., Ng and Cardie (2002b), Uryupina (2003), Poesio et al. (2004b)). Compared to earlier work on anaphoricity determination, recently proposed approaches are more "global" in nature, taking into account the pairwise decisions made by the mention-pair model when making anaphoricity decisions. Examples of such approaches have exploited techniques including integer linear programming (ILP) (Denis and Baldridge, 2007a), label propagation (Zhou and Kong, 2009), and minimum cuts (Ng, 2009).

3.1.5 Combining Classification & Clustering

From a learning perspective, a two-step approach to coreference — classification and clustering — is undesirable. Since the classification model is trained independently of the clustering algorithm, improvements in classification accuracy do not guarantee corresponding improvements in clustering-level accuracy. That is, overall performance on the coreference task might not improve.

To address this problem, McCallum and Wellner (2004) and Finley and Joachims (2005) eliminate the classification step entirely, treating coref-

erence as a *supervised clustering* task where a similarity metric is learned to directly maximize clustering accuracy. Klenner (2007) and Finkel and Manning (2008) use ILP to ensure that the pairwise classification decisions satisfy transitivity.[6]

3.1.6 Weaknesses of the Mention-Pair Model

While many of the aforementioned algorithms for clustering and anaphoricity determination have been shown to improve coreference performance, the underlying model with which they are used in combination — the mention-pair model — remains fundamentally weak. The model has two commonly-cited weaknesses. First, since each candidate antecedent for an anaphoric NP to be resolved is considered independently of the others, the model only determines how good a candidate antecedent is relative to the anaphoric NP, but not how good a candidate antecedent is relative to other candidates. In other words, it fails to answer the question of which candidate antecedent is most probable. Second, it has limitations in its expressiveness: the information extracted from the two NPs alone may not be sufficient for making an informed coreference decision, especially if the candidate antecedent is a pronoun (which is semantically empty) or a mention that lacks descriptive information such as gender (e.g., "Clinton"). Below we discuss how these weaknesses are addressed by the entity-mention model and ranking models.

3.2 Entity-Mention Model

The entity-mention model addresses the expressiveness problem with the mention-pair model. To motivate the entity-mention model, consider an example taken from McCallum and Wellner (2003), where a document consists of three NPs: "Mr. Clinton," "Clinton," and "she." The mention-pair model may determine that "Mr. Clinton" and "Clinton" are coreferent using string-matching features, and that "Clinton" and "she" are coreferent based on proximity and lack of evidence for gender and number disagreement. However, these two pairwise decisions together with transitivity imply that "Mr. Clinton" and "she" will end up in the same cluster, which is incorrect due to gender mismatch. This kind of error arises in part because the later coreference decisions are not dependent on the earlier ones. In particular, had the model taken into consideration that "Mr. Clinton" and "Clinton" were in the same cluster, it probably would not have posited that "she" and "Clinton" are coreferent. The aforementioned Cardie and Wagstaff algorithm attempts to address this problem in a *heuristic* manner. It would be desirable to *learn* a model that can classify whether an NP to be resolved is coreferent with a preceding, possibly partially-formed, cluster. This model is commonly known as the entity-mention model.

Since the entity-mention model aims to classify whether an NP is coreferent with a preceding cluster, each of its training instances (1) corresponds to an NP, NP_k, and a preceding cluster, C_j, and (2) is labeled with either POSITIVE or NEGATIVE, depending on whether NP_k should be assigned to C_j. Consequently, we can represent each instance by a set of *cluster-level* features (i.e., features that are defined over an arbitrary subset of the NPs in C_j). A cluster-level feature can be computed from a feature employed by the mention-pair model by applying a logical predicate. For example, given the NUMBER AGREEMENT feature, which determines whether two NPs agree in number, we can apply the ALL predicate to create a cluster-level feature, which has the value YES if NP_k agrees in number with *all* of the NPs in C_j and NO otherwise. Other commonly-used logical predicates for creating cluster-level features include relaxed versions of the ALL predicate, such as MOST, which is true if NP_k agrees in number with more than half of the NPs in C_j, and ANY, which is true as long as NP_k agrees in number with just one of the NPs in C_j. The ability of the entity-mention model to employ cluster-level features makes it more expressive than its mention-pair counterpart.

Despite its improved expressiveness, the entity-mention model has not yielded particularly encouraging results. For example, Luo et al. (2004) apply the ANY predicate to generate cluster-level features for their entity-mention model, which does not perform as well as the mention-pair model. Yang et al. (2004b; 2008a) also investigate the entity-mention model, which produces results that are only marginally better than those of the mention-pair model. However, it appears that they are not fully exploiting the expressiveness of the entity-mention model, as cluster-level features only comprise a small fraction of their features.

Variants of the entity-mention model have been investigated. For example, Culotta et al. (2007) present a first-order logic model that determines

[6] Recently, however, Klenner and Ailloud (2009) have become less optimistic about ILP approaches to coreference.

the probability that an arbitrary set of NPs are all co-referring. Their model resembles the entity-mention model in that it enables the use of cluster-level features. Daumé III and Marcu (2005) propose an online learning model for constructing coreference chains in an incremental fashion, allowing later coreference decisions to be made by exploiting cluster-level features that are computed over the coreference chains created thus far.

3.3 Ranking Models

While the entity-mention model addresses the expressiveness problem with the mention-pair model, it does not address the other problem: failure to identify the most probable candidate antecedent. Ranking models, on the other hand, allow us to determine which candidate antecedent is most probable given an NP to be resolved. Ranking is arguably a more natural reformulation of coreference resolution than classification, as a ranker allows all candidate antecedents to be considered *simultaneously* and therefore directly captures the competition among them. Another desirable consequence is that there exists a natural resolution strategy for a ranking approach: an anaphoric NP is resolved to the candidate antecedent that has the highest rank. This contrasts with classification-based approaches, where many clustering algorithms have been employed to coordinate the pairwise classification decisions, and it is still not clear which of them is the best.

The notion of ranking candidate antecedents can be traced back to centering algorithms, many of which use grammatical roles to rank forward-looking centers (see Walker et al. (1998)). Ranking is first applied to learning-based coreference resolution by Connolly et al. (1994; 1997), where a model is trained to rank two candidate antecedents. Each training instance corresponds to the NP to be resolved, NP_k, as well as two candidate antecedents, NP_i and NP_j, one of which is an antecedent of NP_k and the other is not. Its class value indicates which of the two candidates is better. This model is referred to as the *tournament* model by Iida et al. (2003) and the *twin-candidate* model by Yang et al. (2003; 2008b). To resolve an NP during testing, one way is to apply the model to each pair of its candidate antecedents, and the candidate that is classified as better the largest number of times is selected as its antecedent.

Advances in machine learning have made it possible to train a *mention ranker* that ranks *all* of the candidate antecedents simultaneously. While mention rankers have consistently outperformed the mention-pair model (Versley, 2006; Denis and Baldridge, 2007b), they are not more expressive than the mention-pair model, as they are unable to exploit cluster-level features, unlike the entity-mention model. To enable rankers to employ cluster-level features, Rahman and Ng (2009) propose the cluster-ranking model, which ranks preceding *clusters*, rather than candidate antecedents, for an NP to be resolved. Cluster rankers therefore address both weaknesses of the mention-pair model, and have been shown to improve mention rankers. Cluster rankers are conceptually similar to Lappin and Leass's (1994) heuristic pronoun resolver, which resolves an anaphoric pronoun to the most salient preceding cluster.

An important issue with ranking models that we have eluded so far concerns the identification of non-anaphoric NPs. As a ranker simply imposes a ranking on candidate antecedents or preceding clusters, it cannot determine whether an NP is anaphoric (and hence should be resolved). To address this problem, Denis and Baldridge (2008) apply an independently trained anaphoricity classifier to identify non-anaphoric NPs prior to ranking, and Rahman and Ng (2009) propose a model that jointly learns coreference and anaphoricity.

4 Knowledge Sources

Another thread of supervised coreference research concerns the development of linguistic features. Below we give an overview of these features.

String-matching features can be computed robustly and typically contribute a lot to the performance of a coreference system. Besides simple string-matching operations such as exact string match, substring match, and head noun match for different kinds of NPs (see Daumé III and Marcu (2005)), slightly more sophisticated string-matching facilities have been attempted, including minimum edit distance (Strube et al., 2002) and longest common subsequence (Castaño et al., 2002). Yang et al. (2004a) treat the two NPs involved as two bags of words, and compute their similarity using metrics commonly-used in information retrieval, such as the dot product, with each word weighted by their TF-IDF value.

Syntactic features are computed based on a syntactic parse tree. Ge et al. (1998) implement

a *Hobbs distance* feature, which encodes the rank assigned to a candidate antecedent for a pronoun by Hobbs's (1978) seminal syntax-based pronoun resolution algorithm. Luo and Zitouni (2005) extract features from a parse tree for implementing Binding Constraints (Chomsky, 1988). Given an automatically parsed corpus, Bergsma and Lin (2006) extract from each parse tree a dependency path, which is represented as a sequence of nodes and dependency labels connecting a pronoun and a candidate antecedent, and collect statistical information from these paths to determine the likelihood that a pronoun and a candidate antecedent connected by a given path are coreferent. Rather than deriving features from parse trees, Iida et al. (2006) and Yang et al. (2006) employ these trees directly as *structured* features for pronoun resolution. Specifically, Yang et al. define tree kernels for efficiently computing the similarity between two parse trees, and Iida et al. use a boosting-based algorithm to compute the usefulness of a subtree.

Grammatical features encode the grammatical properties of one or both NPs involved in an instance. For example, Ng and Cardie's (2002c) resolver employs 34 grammatical features. Some features determine NP type (e.g., are both NPs definite or pronouns?). Some determine the grammatical role of one or both of the NPs. Some encode traditional linguistic (hard) constraints on coreference. For example, coreferent NPs have to agree in number and gender and cannot span one another (e.g., "Google" and "Google employees"). There are also features that encode general linguistic preferences either for or against coreference. For example, an indefinite NP (that is not in apposition to an anaphoric NP) is not likely to be coreferent with any NP that precedes it.

There has been an increasing amount of work on investigating **semantic features** for coreference resolution. One of the earliest kinds of semantic knowledge employed for coreference resolution is perhaps selectional preference (Dagan and Itai, 1990; Kehler et al., 2004b; Yang et al., 2005; Haghighi and Klein, 2009): given a pronoun to be resolved, its governing verb, and its grammatical role, we prefer a candidate antecedent that can be governed by the same verb and be in the same role. Semantic knowledge has also been extracted from WordNet and unannotated corpora for computing the semantic compatibility/similarity between two common nouns (Harabagiu et al., 2001; Versley, 2007) as well as the semantic class of a noun (Ng, 2007a; Huang et al., 2009). One difficulty with deriving knowledge from WordNet is that one has to determine which sense of a given word to use. Some researchers simply use the first sense (Soon et al., 2001) or all possible senses (Ponzetto and Strube, 2006a), while others overcome this problem with word sense disambiguation (Nicolae and Nicolae, 2006). Knowledge has also been mined from Wikipedia for measuring the semantic relatedness of two NPs, NP_j and NP_k (Ponzetto and Strube (2006a; 2007)), such as: whether $\text{NP}_{j/k}$ appears in the first paragraph of the Wiki page that has $\text{NP}_{k/j}$ as the title or in the list of categories to which this page belongs, and the degree of overlap between the two pages that have the two NPs as their titles (see Poesio et al. (2007) for other uses of encyclopedic knowledge for coreference resolution). Contextual roles (Bean and Riloff, 2004), semantic relations (Ji et al., 2005), semantic roles (Ponzetto and Strube, 2006b; Kong et al., 2009), and animacy (Orăsan and Evans, 2007) have also been exploited to improve coreference resolution.

Lexico-syntactic *patterns* have been used to capture the semantic relatedness between two NPs and hence the likelihood that they are coreferent. For instance, given the pattern *X is a Y* (which is highly indicative that *X* and *Y* are coreferent), we can instantiate it with a pair of NPs and search for the instantiated pattern in a large corpus or the Web (Daumé III and Marcu, 2005; Haghighi and Klein, 2009). The more frequently the pattern occurs, the more likely they are coreferent. This technique has been applied to resolve different kinds of anaphoric references, including *other*-anaphora (Modjeska et al., 2003; Markert and Nissim, 2005) and bridging references (Poesio et al., 2004a). While these patterns are typically hand-crafted (e.g., Garera and Yarowsky (2006)), they can also be learned from an annotated corpus (Yang and Su, 2007) or bootstrapped from an unannotated corpus (Bean and Riloff, 2004).

Despite the large amount of work on discourse-based anaphora resolution in the 1970s and 1980s (see Hirst (1981)), learning-based resolvers have only exploited shallow **discourse-based features**, which primarily involve characterizing the salience of a candidate antecedent by measuring its distance from the anaphoric NP to be resolved or determining whether it is in a prominent grammatical role (e.g., subject). A notable exception

is Iida et al. (2009), who train a ranker to rank the candidate antecedents for an anaphoric pronoun by their salience. It is worth noting that Tetreault (2005) has employed Grosz and Sidner's (1986) discourse theory and Veins Theory (Ide and Cristea, 2000) to identify and remove candidate antecedents that are not referentially accessible to an anaphoric pronoun in his heuristic pronoun resolvers. It would be interesting to incorporate this idea into a learning-based resolver.

There are also features that do not fall into any of the preceding categories. For example, a memorization feature is a word pair composed of the head nouns of the two NPs involved in an instance (Bengtson and Roth, 2008). Memorization features have been used as binary-valued features indicating the presence or absence of their words (Luo et al., 2004) or as probabilistic features indicating the probability that the two heads are coreferent according to the training data (Ng, 2007b). An anaphoricity feature indicates whether an NP to be resolved is anaphoric, and is typically computed using an anaphoricity classifier (Ng, 2004), hand-crafted patterns (Daumé III and Marcu, 2005), and automatically acquired patterns (Bean and Riloff, 1999). Finally, the outputs of rule-based pronoun and coreference resolvers have also been used as features for learning-based coreference resolution (Ng and Cardie, 2002c).

For an empirical evaluation of the contribution of a subset of these features to the mention-pair model, see Bengtson and Roth (2008).

5 Evaluation Issues

Two important issues surround the evaluation of a coreference resolver. First, how do we obtain the set of NPs that a resolver will partition? Second, how do we score the partition it produces?

5.1 Extracting Candidate Noun Phrases

To obtain the set of NPs to be partitioned by a resolver, three methods are typically used. In the first method, the NPs are extracted automatically from a syntactic parser. The second method involves extracting the NPs directly from the gold standard. In the third method, a *mention detector* is first trained on the gold-standard NPs in the training texts, and is then applied to automatically extract *system mentions* in a test text.[7] Note that these three extraction methods typically produce different numbers of NPs: the NPs extracted from a parser tend to significantly outnumber the system mentions, which in turn outnumber the gold NPs. The reasons are two-fold. First, in some coreference corpora (e.g., MUC-6 and MUC-7), the NPs that are not part of any coreference chain are not annotated. Second, in corpora such as those produced by the ACE evaluations, only the NPs that belong to one of the ACE entity types (e.g., PERSON, ORGANIZATION, LOCATION) are annotated.

Owing in large part to the difference in the number of NPs extracted by these three methods, a coreference resolver can produce substantially different results when applied to the resulting three sets of NPs, with gold NPs yielding the best results and NPs extracted from a parser yielding the worst (Nicolae and Nicolae, 2006). While researchers who evaluate their resolvers on gold NPs point out that the results can more accurately reflect the performance of their coreference algorithm, Stoyanov et al. (2009) argue that such evaluations are unrealistic, as NP extraction is an integral part of an end-to-end fully-automatic resolver.

Whichever NP extraction method is employed, it is clear that the use of gold NPs can considerably simplify the coreference task, and hence resolvers employing different extraction methods should *not* be compared against each other.

5.2 Scoring a Coreference Partition

The MUC scorer (Vilain et al., 1995) is the first program developed for scoring coreference partitions. It has two often-cited weaknesses. As a *link-based* measure, it does not reward correctly identified singleton clusters since there is no coreference link in these clusters. Also, it tends to underpenalize partitions with overly large clusters.

To address these problems, two coreference scoring programs have been developed: B^3 (Bagga and Baldwin, 1998) and CEAF (Luo, 2005). Note that both scorers have only been defined for the case where the key partition has the same set of NPs as the response partition. To apply these scorers to automatically extracted NPs, different methods have been proposed (see Rahman and Ng (2009) and Stoyanov et al. (2009)).

Since coreference is a clustering task, any general-purpose method for evaluating a response partition against a key partition (e.g., Kappa (Carletta, 1996)) can be used for coreference scor-

[7] An exception is Daumé III and Marcu (2005), whose model jointly learns to extract NPs and perform coreference.

ing (see Popescu-Belis et al. (2004)). In practice, these general-purpose methods are typically used to provide scores that complement those obtained via the three coreference scorers discussed above. It is worth mentioning that there is a trend towards evaluating a resolver against multiple scorers, which can indirectly help to counteract the bias inherent in a particular scorer. For further discussion on evaluation issues, see Byron (2001).

6 Concluding Remarks

While we have focused our discussion on supervised approaches, coreference researchers have also attempted to reduce a resolver's reliance on annotated data by combining a small amount of labeled data and a large amount of unlabeled data using general-purpose semi-supervised learning algorithms such as co-training (Müller et al., 2002), self-training (Kehler et al., 2004a), and EM (Cherry and Bergsma, 2005; Ng, 2008). Interestingly, recent results indicate that unsupervised approaches to coreference resolution (e.g., Haghighi and Klein (2007; 2010), Poon and Domingos (2008)) rival their supervised counterparts, casting doubts on whether supervised resolvers are making effective use of the available labeled data.

Another issue that we have not focused on but which is becoming increasingly important is multilinguality. While many of the techniques discussed in this paper were originally developed for English, they have been applied to learn coreference models for other languages, such as Chinese (e.g., Converse (2006)), Japanese (e.g., Iida (2007)), Arabic (e.g., Luo and Zitouni (2005)), Dutch (e.g., Hoste (2005)), German (e.g., Wunsch (2010)), Swedish (e.g., Nilsson (2010)), and Czech (e.g., Nguỵ et al. (2009)). In addition, researchers have developed approaches that are targeted at handling certain kinds of anaphora present in non-English languages, such as zero anaphora (e.g., Iida et al. (2007a), Zhao and Ng (2007)).

As Mitkov (2001) puts it, coreference resolution is a "difficult, but not intractable problem," and we have been making "slow, but steady progress" on improving machine learning approaches to the problem in the past fifteen years. To ensure further progress, researchers should compare their results against a baseline that is stronger than the commonly-used Soon et al. (2001) system, which relies on a weak model (i.e., the mention-pair model) and a small set of linguistic features. As recent systems are becoming more sophisticated, we suggest that researchers make their systems publicly available in order to facilitate performance comparisons. Publicly available coreference systems currently include JavaRAP (Qiu et al., 2004), GuiTaR (Poesio and Kabadjov, 2004), BART (Versley et al., 2008b), CoRTex (Denis and Baldridge, 2008), the Illinois Coreference Package (Bengtson and Roth, 2008), CherryPicker (Rahman and Ng, 2009), Reconcile (Stoyanov et al., 2010), and Charniak and Elsner's (2009) pronoun resolver.

We conclude with a discussion of two questions regarding supervised coreference research. First, *what is the state of the art?* This is not an easy question, as researchers have been evaluating their resolvers on different corpora using different evaluation metrics and preprocessing tools. In particular, preprocessing tools can have a large impact on the performance of a resolver (Barbu and Mitkov, 2001). Worse still, assumptions about whether gold or automatically extracted NPs are used are sometimes not explicitly stated, potentially causing results to be interpreted incorrectly. To our knowledge, however, the best results on the MUC-6 and MUC-7 data sets using automatically extracted NPs are reported by Yang et al. (2003) (71.3 MUC F-score) and Ng and Cardie (2002c) (63.4 MUC F-score), respectively;[8] and the best results on the ACE data sets using gold NPs can be found in Luo (2007) (88.4 ACE-value).

Second, *what lessons can we learn from fifteen years of learning-based coreference research?* The mention-pair model is weak because it makes coreference decisions based on local information (i.e., information extracted from two NPs). Expressive models (e.g., those that can exploit cluster-level features) generally offer better performance, and so are models that are "global" in nature. Global coreference models may refer to any kind of models that can exploit non-local information, including models that can consider multiple candidate antecedents simultaneously (e.g., ranking models), models that allow joint learning for coreference resolution and related tasks (e.g., anaphoricity determination), models that can directly optimize clustering-level (rather than classification) accuracy, and models that can coordinate with other components of a resolver, such as training instance creation and clustering.

[8] These results by no means suggest that no progress has been made since 2003: most of the recently proposed coreference models were evaluated on the ACE data sets.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views or official policies, either expressed or implied, of the NSF.

References

Chinatsu Aone and Scott William Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 122–129.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC Workshop on Linguistic Coreference*, pages 563–566.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2002. Correlation clustering. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 238–247.

Catalina Barbu and Ruslan Mitkov. 2001. Evaluation tool for rule-based anaphora resolution methods. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 34–41.

David Bean and Ellen Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 373–380.

David Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Human Language Technologies 2004: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 297–304.

Eric Bengtson and Dan Roth. 2008. Understanding the values of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 294–303.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2008. Distributional identification of non-referential pronouns. In *Proceedings of ACL-08: HLT*, pages 10–18.

Donna Byron. 2001. The uncommon denominator: A proposal for consistent reporting of pronoun resolution results. *Computational Linguistics*, 27(4):569–578.

Sasha Calhoun, Jean Carletta, Jason Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. (in press). The NXT-format Switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation*.

Claire Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254.

José Castaño, Jason Zhang, and James Pustejovsky. 2002. Anaphora resolution in biomedical literature. In *Proceedings of the 2002 International Symposium on Reference Resolution*.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 148–156.

Eugene Charniak. 1972. *Towards a Model of Children's Story Comphrension*. AI-TR 266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA.

Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 88–95.

Noam Chomsky. 1988. *Language and Problems of Knowledge. The Managua Lectures*. MIT Press, Cambridge, Massachusetts.

William Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123.

Dennis Connolly, John D. Burger, and David S. Day. 1994. A machine learning approach to anaphoric reference. In *Proceedings of International Conference on New Methods in Language Processing*, pages 255–261.

Dennis Connolly, John D. Burger, and David S. Day. 1997. A machine learning approach to anaphoric reference. In D. Jones and H. Somers, editors, *New Methods in Language Processing*, pages 133–144. UCL Press.

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. thesis, University of Pennsylvania, USA.

Aron Culotta, Michael Wick, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88.

Walter Daelemans and Antal Van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press, Cambridge, UK.

Ido Dagan and Alon Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 330–332.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 97–104.

Arthur Dempster. 1968. A generalization of Bayesian inference. *Journal of the Royal Statistical Society*, 30:205–247.

Pascal Denis and Jason Baldridge. 2007a. Global, joint determination of anaphoricity and coreference resolution using integer programming. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 236–243.

Pascal Denis and Jason Baldridge. 2007b. A ranking approach to pronoun resolution. In *Proceedings of the Twentieth International Conference on Artificial Intelligence*, pages 1588–1593.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669.

Richard Evans. 2001. Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 16(1):45–57.

Jenny Rose Finkel and Christopher Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-08: HLT, Short Papers*, pages 45–48.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 217–224.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Nikesh Garera and David Yarowsky. 2006. Resolving and generating definite anaphora by modeling hypernymy using unlabeled corpora. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 37–44.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170.

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

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1983. Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 44–50.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226.

Barbara J. Grosz. 1977. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 67–76.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 848–855.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1152–1161.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Stěpánek, Jiří Havelka, and Marie Mikulová. 2006. The Prague Dependency Treebank 2.0. In *Linguistic Data Consortium*.

Sanda Harabagiu, Răzvan Bunescu, and Steven Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 55–62.

Laura Hasler, Constantin Orasan, and Karin Naumann. 2006. NPs for events: Experiments in coreference annotation. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1167–1172.

Peter Heeman and James Allen. 1995. The TRAINS spoken dialog corpus. CD-ROM, Linguistic Data Consortium.

Graeme Hirst. 1981. Discourse-oriented anaphora resolution in natural language understanding: A review. *American Journal of Computational Linguistics*, 7(2):85–98.

Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

Véronique Hoste and Walter Daelemans. 2005. Comparing learning approaches to coreference resolution. There is more to it than bias. In *Proceedings of the ICML Workshop on Meta-Learning*.

Véronique Hoste. 2005. *Optimization Issues in Machine Learning of Coreference Resolution*. Ph.D. thesis, University of Antewerp, Belgium.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Zhiheng Huang, Guangping Zeng, Weiqun Xu, and Asli Celikyilmaz. 2009. Accurate semantic class classifier for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1232–1240.

Nancy Ide and Dan Cristea. 2000. A hierarchical account of referential accessibility. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 416–424.

Ryu Iida, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 625–632.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007a. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing*, 6(4).

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007b. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the ACL Workshop 'Linguistic Annotation Workshop'*, pages 132–139.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2009. Capturing salience with a trainable cache model for zero-anaphora resolution. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 647–655.

Ryu Iida. 2007. *Combining Linguistic Knowledge and Machine Learning for Anaphora Resolution*. Ph.D. thesis, Nara Institute of Science and Technology, Japan.

Heng Ji, David Westbrook, and Ralph Grishman. 2005. Using semantic relations to refine coreference decisions. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 17–24.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004a. Competitive self-trained pronoun interpretation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 33–36.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004b. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Human Language Technologies 2004: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 289–296.

Andrew Kehler. 1997. Probabilistic coreference in information extraction. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 163–173.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphor for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 113–118.

Manfred Klenner and Étienne Ailloud. 2008. Enhancing coreference clustering. In *Proceedings of the Second Workshop on Anaphora Resolution*, pages 31–40.

Manfred Klenner and Étienne Ailloud. 2009. Optimization in coreference resolution is not needed: A nearly-optimal algorithm with intensional constraints. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 442–450.

Manfred Klenner. 2007. Enforcing consistency on coreference sets. In *Proceedings of Recent Advances in Natural Language Processing*.

Fang Kong, GuoDong Zhou, and Qiaoming Zhu. 2009. Employing the centering theory in pronoun resolution from the semantic perspective. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 987–996.

Shalom Lappin and Herbert Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–562.

Xiaoqiang Luo and Imed Zitouni. 2005. Multi-lingual coreference resolution with syntactic features. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 660–667.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 135–142.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Xiaoqiang Luo. 2007. Coreference or not: A twin model for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 73–80.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Katja Markert and Malvina Nissim. 2005. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367–402.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI Workshop on Information Integration on the Web*.

Andrew McCallum and Ben Wellner. 2004. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Proceesing Systems*.

Joseph McCarthy and Wendy Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 1050–1055.

Ruslan Mitkov. 1999. Anaphora resolution: The state of the art. Technical Report (Based on the COLING/ACL-98 tutorial on anaphora resolution), University of Wolverhampton, Wolverhampton.

Ruslan Mitkov. 2001. Outstanding issues in anaphora resolution. In Al. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 110–125. Springer.

Ruslan Mitkov. 2002. *Anaphora Resolution*. Longman.

Natalia N. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 176–183.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference*.

MUC-7. 1998. *Proceedings of the Seventh Message Understanding Conference*.

Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 352–359.

Christoph Müller. 2006. Automatic detection of non-referential it in spoken multi-party dialog. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–56.

Vincent Ng and Claire Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 55–62.

Vincent Ng and Claire Cardie. 2002b. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 730–736.

Vincent Ng and Claire Cardie. 2002c. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Vincent Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 151–158.

Vincent Ng. 2007a. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 536–543.

Vincent Ng. 2007b. Shallow semantics for coreference resolution. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1689–1694.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 640–649.

Vincent Ng. 2009. Graph-cut-based anaphoricity determination for coreference resolution. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 575–583.

Giang Linh Ngụy, Václav Novák, and Zdeněk Žabokrtský. 2009. Comparison of classification and ranking approaches to pronominal anaphora resolution in Czech. In *Proceedings of the SIGDIAL 2009 Conference*, pages 276–285.

Cristina Nicolae and Gabriel Nicolae. 2006. BestCut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 275–283.

Kristina Nilsson. 2010. *Hybrid Methods for Coreference Resolution in Swedish*. Ph.D. thesis, Stockholm University, Sweden.

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

Constantin Orăsan and Richard Evans. 2007. NP animacy identification for anaphora resolution. *Journal of Artificial Intelligence Research*, 29:79 – 103.

Constantin Orăsan, Dan Cristea, Ruslan Mitkov, and António H. Branco. 2008. Anaphora Resolution Exercise: An overview. In *Proceedings of the 6th Language Resources and Evaluation Conference*, pages 2801–2805.

Chris Paice and Gareth Husk. 1987. Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun 'it'. *Computer Speech and Language*, 2:109–132.

Massimo Poesio and Mijail A. Kabadjov. 2004. A general-purpose, off-the-shelf anaphora resolution module: Implementation and preliminary evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 663–668.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004a. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 143–150.

Massimo Poesio, Olga Uryupina, Renata Vieira, Mijail Alexandrov-Kabadjov, and Rodrigo Goulart. 2004b. Discourse-new detectors for definite description resolution: A survey and a preliminary proposal. In *Proeedings of the ACL Workshop on Reference Resolution*.

Massimo Poesio, David Day, Ron Artstein, Jason Duncan, Vladimir Eidelman, Claudio Giuliano, Rob Hall, Janet Hitzeman, Alan Jern, Mijail Kabadjov, Stanley Yong Wai Keong, Gideon Mann, Alessandro Moschitti, Simone Ponzetto, Jason Smith, Josef Steinberger, Michael Strube, Jian Su, Yannick Versley, Xiaofeng Yang, and Michael Wick. 2007. ELERFED: Final report of the research group on Exploiting Lexical and Encyclopedic Resources For Entity Disambiguation. Technical report, Summer Workshop on Language Engineering, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD.

Simone Paolo Ponzetto and Massimo Poesio. 2009. State-of-the-art NLP approaches to coreference resolution: Theory and practical recipes. In *Tutorial Abstracts of ACL-IJCNLP 2009*, page 6.

Simone Paolo Ponzetto and Michael Strube. 2006a. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Human Language Technologies 2006: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 192–199.

Simone Paolo Ponzetto and Michael Strube. 2006b. Semantic role labeling for coreference resolution. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 143–146.

Simone Paolo Ponzetto and Michael Strube. 2007. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 650–659.

Andrei Popescu-Belis, Loïs Rigouste, Susanne Salmon-Alt, and Laurent Romary. 2004. Online evaluation of coreference resolution. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1507–1510.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2004. A public reference implementation of the RAP anaphora resolution algorithm. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 291–294.

John Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Altaf Rahman and Vincent Ng. 2009. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977.

Marta Recasens and M. Antónia Martí. 2009. AnCora-CO: Coreferentially annotated corpora for Spanish and Catalan. *Language Resources and Evaluation*, 43(4).

Marta Recasens, Toni Martí, Mariona Taulé, Lluís Màrquez, and Emili Sapena. 2009. SemEval-2010 Task 1: Coreference resolution in multiple languages. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions* (SEW-2009), pages 70–75.

Candace Sidner. 1979. *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse*. Ph.D. thesis, Massachusetts Institute of Technology, USA.

Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. 1999. Corpus-based learning for noun phrase coreference resolution. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 285–291.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 656–664.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Coreference resolution with Reconcile. In *Proceedings of the ACL 2010 Conference Short Papers*.

Michael Strube, Stefan Rapp, and Christoph Müller. 2002. The influence of minimum edit distance on reference resolution. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 312–319.

Michael Strube. 2002. NLP approaches to reference resolution. In *Tutorial Abstracts of ACL 2002*, page 124.

Michael Strube. 2009. Anaphernresolution. In *Computerlinguistik und Sprachtechnologie. Eine Einfuhrung*. Springer, Heidelberg, Germany, 3rd edition.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The tüba-d/z treebank: Annotating German with a context-free backbone. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 2229–2235.

Joel Tetreault. 2005. *Empirical Evaluations of Pronoun Resolution*. Ph.D. thesis, University of Rochester, USA.

Olga Uryupina. 2003. High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL Student Research Workshop*, pages 80–86.

Olga Uryupina. 2004. Linguistically motivated sample selection for coreference resolution. In *Proceedings of the 5th Discourse Anaphora and Anaphor Resolution Colloquium*.

Kees van Deemter and Rodger Kibble. 2000. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008a. Coreference systems based on kernels methods. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 961–968.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008b. BART: A modular toolkit for coreference resolution. In *Proceedings of the ACL-08: HLT Demo Session*, pages 9–12.

Yannick Versley. 2006. A constraint-based approach to noun phrase coreference resolution in German newspaper text. In *Konferenz zur Verarbeitung Natürlicher Sprache*.

Yannick Versley. 2007. Antecedent selection techniques for high-recall coreference resolution. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 496–505.

Renata Vieira and Massimo Poesio. 2000. Processing definite descriptions in corpora. In S. Botley and A. McEnery, editors, *Corpus-based and Computational Approaches to Discourse Anaphora*, pages 189–212. UCL Press.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*, pages 45–52.

Marilyn Walker, Aravind Joshi, and Ellen Prince, editors. 1998. *Centering Theory in Discourse*. Oxford University Press.

Holger Wunsch. 2010. *Rule-based and Memory-based Pronoun Resolution for German: A Comparison and Assessment of Data Sources*. Ph.D. thesis, University of Tübingen, Germany.

Xiaofeng Yang and Jian Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 528–535.

Xiaofeng Yang, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competitive learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2004a. Improving noun phrase coreference resolution by matching strings. In *Proceedings of the First International Joint Conference on Natural Language Processing*, pages 22–31.

Xiaofeng Yang, Jian Su, GuoDong Zhou, and Chew Lim Tan. 2004b. An NP-cluster based approach to coreference resolution. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 226–232.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 165–172.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2006. Kernel based pronoun resolution with structured syntactic knowledge. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 41–48.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, and Sheng Li. 2008a. An entity-mention model for coreference resolution with inductive logic programming. In *Proceedings of ACL-08: HLT*, pages 843–851.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2008b. A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356.

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbetts. 2004. Coreference resolution for information extraction. In *Proceedings of the ACL Workshop on Reference Resolution and its Applications*, pages 9–16.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*, pages 541–550.

GuoDong Zhou and Fang Kong. 2009. Global learning of noun phrase anaphoricity in coreference resolution via label propagation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 978–986.

Unsupervised Event Coreference Resolution with Rich Linguistic Features

Cosmin Adrian Bejan
Institute for Creative Technologies
University of Southern California
Marina del Rey, CA 90292, USA

Sanda Harabagiu
Human Language Technology Institute
University of Texas at Dallas
Richardson, TX 75083, USA

Abstract

This paper examines how a new class of nonparametric Bayesian models can be effectively applied to an open-domain event coreference task. Designed with the purpose of clustering complex linguistic objects, these models consider a potentially infinite number of features and categorical outcomes. The evaluation performed for solving both within- and cross-document event coreference shows significant improvements of the models when compared against two baselines for this task.

1 Introduction

The event coreference task consists of finding clusters of event mentions that refer to the same event. Although it has not been extensively studied in comparison with the related problem of entity coreference resolution, solving event coreference has already proved its usefulness in various applications such as topic detection and tracking (Allan et al., 1998), information extraction (Humphreys et al., 1997), question answering (Narayanan and Harabagiu, 2004), textual entailment (Haghighi et al., 2005), and contradiction detection (de Marneffe et al., 2008).

Previous approaches for solving event coreference relied on supervised learning methods that explore various linguistic properties in order to decide if a pair of event mentions is coreferential or not (Humphreys et al., 1997; Bagga and Baldwin, 1999; Ahn, 2006; Chen and Ji, 2009). In spite of being successful for a particular labeled corpus, these pairwise models are dependent on the domain or language that they are trained on. Moreover, since event coreference resolution is a complex task that involves exploring a rich set of linguistic features, annotating a large corpus with event coreference information for a new language or domain of interest requires a substantial amount of manual effort. Also, since these models are dependent on local pairwise decisions, they are unable to capture a global event distribution at topic or document collection level.

To address these limitations and to provide a more flexible representation for modeling observable data with rich properties, we present two novel, fully generative, nonparametric Bayesian models for unsupervised within- and cross-document event coreference resolution. The first model extends the *hierarchical Dirichlet process* (Teh et al., 2006) to take into account additional properties associated with observable objects (i.e., event mentions). The second model overcomes some of the limitations of the first model. It uses the *infinite factorial hidden Markov model* (Van Gael et al., 2008b) coupled to the *infinite hidden Markov model* (Beal et al., 2002) in order to (1) consider a potentially infinite number of features associated with observable objects, (2) perform an automatic selection of the most salient features, and (3) capture the structural dependencies of observable objects at the discourse level. Furthermore, both models are designed to account for a potentially infinite number of categorical outcomes (i.e., events). These models provide additional details and experimental results to our preliminary work on unsupervised event coreference resolution (Bejan et al., 2009).

2 Event Coreference

The problem of determining if two events are identical was originally studied in philosophy. One relevant theory on event identity was proposed by Davidson (1969) who argued that two events are identical if they have the same causes and effects. Later on, a different theory was proposed by Quine (1985) who considered that each event refers to a physical object (which is well defined in space and time), and therefore, two events are identical

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1412–1422,
Uppsala, Sweden, 11-16 July 2010.

if they have the same spatiotemporal location. In (Davidson, 1985), Davidson abandoned his suggestion to embrace the Quinean theory on event identity (Malpas, 2009).

2.1 An Example

In accordance with the Quinean theory, we consider that two event mentions are coreferential if they have the same *event properties* and share the same *event participants*. For instance, the sentences from Example 1 encode event mentions that refer to several individuated events. These sentences are extracted from a newly annotated corpus with event coreference information (see Section 4). In this corpus, we organize documents that describe the same seminal event into topics. In particular, the topics shown in this example describe the seminal event of buying ATI by AMD (topic 43) and the seminal event of buying EDS by HP (topic 44).

Although all the event mentions of interest emphasized in boldface in Example 1 evoke the same generic event *buy*, they refer to three individuated events: $e_1 = \{em_1, em_2\}$, $e_2 = \{em_{3-6}, em_8\}$, and $e_3 = \{em_7\}$. For example, em_1(*buy*) and em_3(*buy*) correspond to different individuated events since they have a different AGENT ([BUYER(em_1)=*AMD*] $\neq$ [BUYER(em_3)=*HP*]). This organization of event mentions leads to the idea of creating an event hierarchy which has on the first level, *event mentions*, on the second level, *individuated events*, and on the third level, *generic events*. In particular, the event hierarchy corresponding to the event mentions annotated in our example is illustrated in Figure 1.

Solving the event coreference problem poses many interesting challenges. For instance, in order to solve the coreference chain of event mentions that refer to the event e_2, we need to take into account the following issues: (i) a coreference chain can encode both within- and cross-document coreference information; (ii) two mentions from the same chain can have different word classes (e.g., em_3(*buy*)–verb, em_4(*purchase*)–noun); (iii) not all the mentions from the same chain are synonymous (e.g., em_3(*buy*) and em_8(*acquire*)), although a semantic relation might exist between them (e.g., in WordNet (Fellbaum, 1998), the genus of *buy* is *acquire*); (iv) partial (or all) properties and participants of an event mention can be omitted in text (e.g., em_4(*purchase*)). In Section 5, we discuss additional aspects of the event coreference problem that are not revealed in Example 1.

Topic 43

Document 3

s$_4$: *AMD agreed to* [***buy***]$_{em_1}$ *Markham, Ontario-based ATI for around $5.4 billion in cash and stock, the companies announced Monday.*

s$_5$: *The* [***acquisition***]$_{em_2}$ *would turn AMD into one of the world's largest providers of graphics chips.*

Topic 44

Document 2

s$_1$: *Hewlett-Packard is negotiating to* [***buy***]$_{em_3}$ *technology services provider Electronic Data Systems.*

s$_8$: *With a market value of about $115 billion, HP could easily use its own stock to finance the* [***purchase***]$_{em_4}$.

s$_9$: *If the* [***deal***]$_{em_5}$ *is completed, it would be HP's biggest* [***acquisition***]$_{em_6}$ *since it* [***bought***]$_{em_7}$ *Compaq Computer Corp. for $19 billion in 2002.*

Document 5

s$_2$: *Industry sources have confirmed to eWEEK that Hewlett-Packard will* [***acquire***]$_{em_8}$ *Electronic Data Systems for about $13 billion.*

Example 1: Examples of event mention annotations.

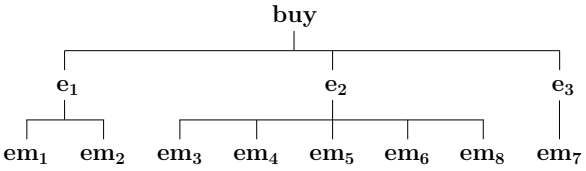


Figure 1: Fragment from the event hierarchy.

2.2 Linguistic Features

The events representing coreference clusters of event mentions are characterized by a large set of linguistic features. To compute an accurate event distribution for event coreference resolution, we associate the following categories of linguistic features with each annotated event mention.

Lexical Features (LF) We capture the lexical context of an event mention by extracting the following features: the head word (HW), the lemmatized head word (HL), the lemmatized left and right words surrounding the mention (LHL,RHL), and the HL features corresponding to the left and right mentions (LHE,RHE). For instance, the lexical features extracted for the event mention em_7(*bought*) from our example are HW:*bought*, HL:*buy*, LHL:*it*, RHL:*Compaq*, LHE:*acquisition*, and RHE:*acquire*.

Class Features (CF) These features aim to group mentions into several types of classes: the part-of-speech of the HW feature (POS), the word class of the HW feature (HWC), and the event class of the mention (EC). The HWC feature can take one of the following values: VERB, NOUN, ADJEC-

TIVE, and OTHER. As values for the EC feature, we consider the seven event classes defined in the TimeML specification language (Pustejovsky et al., 2003a): OCCURRENCE, PERCEPTION, REPORTING, ASPECTUAL, STATE, I_ACTION, and I_STATE. In order to extract the event classes corresponding to the event mentions from a given dataset, we employed the event extractor described in (Bejan, 2007). This extractor is trained on the TimeBank corpus (Pustejovsky et al., 2003b), which is a TimeML resource encoding temporal elements such as events, time expressions, and temporal relations.

WordNet Features (WF) In our efforts to create clusters of event mention attributes as close as possible to the true attribute clusters of the individuated events, we build two sets of word clusters using the entire lexical information from the WordNet database. After creating these sets of clusters, we then associate each event mention with only one cluster from each set. The first set uses the transitive closure of the WordNet SYNONYMOUS relation to form clusters with all the words from WordNet (WNS). For instance, the verbs *buy* and *purchase* correspond to the same cluster ID because there exist a chain of SYNONYMOUS relations between them in WordNet. The second set considers as grouping criteria the categorization of words from the WordNet lexicographer's files (WNL). In addition, for each word that is not covered in WordNet, we create a new cluster ID in each set of clusters.

Semantic Features (SF) To extract features that characterize participants and properties of event mentions, we use the semantic parser described in (Bejan and Hathaway, 2007). One category of semantic features that we identify for event mentions is the *predicate argument structures* encoded in PropBank annotations (Palmer et al., 2005). In PropBank, the predicate argument structures are represented by events expressed as verbs in text and by the semantic roles, or *predicate arguments*, associated with these events. For example, ARG0 annotates a specific type of semantic role which represents the AGENT, DOER, or ACTOR of a specific event. Another argument is ARG1, which plays the role of the PATIENT, THEME, or EXPERIENCER of an event. In particular, the predicate arguments associated to the event mention em_8(*bought*) from Example 1 are ARG0:[*it*], ARG1:[*Compaq Computer Corp.*], ARG3:[*for $19 billion*], and ARG-TMP:[*in 2002*].

Event mentions are not only expressed as verbs in text, but also as nouns and adjectives. Therefore, for a better coverage of semantic features, we also employ the semantic annotations encoded in the FrameNet corpus (Baker et al., 1998). FrameNet annotates word expressions capable of evoking conceptual structures, or *semantic frames*, which describe specific situations, objects, or events (Fillmore, 1982). The semantic roles associated with a word in FrameNet, or *frame elements*, are locally defined for the semantic frame evoked by the word. In general, the words annotated in FrameNet are expressed as verbs, nouns, and adjectives.

To preserve the consistency of semantic role features, we align frame elements to predicate arguments by running the PropBank semantic parser on the manual annotations from FrameNet; conversely, we also run the FrameNet parser on the manual annotations from PropBank. Moreover, to obtain a better alignment of semantic roles, we run both parsers on a large amount of unlabeled text. The result of this process is a map with all frame elements statistically aligned to all predicate arguments. For instance, in 99.7% of the cases the frame element BUYER of the semantic frame COMMERCE BUY is mapped to ARG0, and in the remaining 0.3% of the cases to ARG1. Additionally, we use this map to create a more general semantic feature which assigns to each predicate argument a frame element label. In particular, the features for em_8(*acquire*) are FEA0:BUYER, FEA1:GOODS, FEA3:MONEY, and FEATMP:TIME.

Two additional semantic features used in our experiments are: (1) the semantic frame (FR) evoked by every mention;[1] and (2) the WNS feature applied to the head word of every semantic role (e.g., WSA0, WSA1).

Feature Combinations (FC) We also explore various combinations of the features presented above. Examples include HW+HWC, HL+FR, FR+ARG1, LHL+RHL, etc.

It is worth noting that there exist event mentions for which not all the features can be extracted. For example, the LHE and RHE features are missing for the first and last event mentions in a document, respectively. Also, many semantic roles can be absent for an event mention in a given context.

[1] The reason for extracting this feature is given by the fact that, in general, frames are able to capture properties of generic events (Lowe et al., 1997).

3 Nonparametric Bayesian Models

As input for our models, we consider a collection of I documents, where each document i has J_i event mentions. For features, we make the distinction between *feature types* and *feature values* (e.g., POS is a feature type and has values such as NN and VB). Each event mention is characterized by L feature types, FT, and each feature type is represented by a finite vocabulary of feature values, fv. Thus, we can represent the observable properties of an event mention as a vector of L feature type – feature value pairs $\langle(\text{FT}_1 : fv_{1i}), \ldots, (\text{FT}_L : fv_{Li})\rangle$, where each feature value index i ranges in the feature value space associated with a feature type.

3.1 A Finite Feature Model

We present an extension of the *hierarchical Dirichlet process* (HDP) model which is able to represent each observable object (i.e., event mention) by a finite number of feature types L. Our HDP extension is also inspired from the Bayesian model proposed by Haghighi and Klein (2007). However, their model is strictly customized for entity coreference resolution, and therefore, extending it to include additional features for each observable object is a challenging task (Ng, 2008; Poon and Domingos, 2008).

In the HDP model, a *Dirichlet process* (DP) (Ferguson, 1973) is associated with each document, and each mixture component (i.e., event) is shared across documents. To describe its extension, we consider $\mathbf{Z}$ the set of indicator random variables for indices of events, ϕ_z the set of parameters associated with an event z, $\boldsymbol{\phi}$ a notation for all model parameters, and $\mathbf{X}$ a notation for all random variables that represent observable features.[2] Given a document collection annotated with event mentions, the goal is to find the best assignment of event indices $\mathbf{Z}^*$, which maximize the posterior probability $P(\mathbf{Z}|\mathbf{X})$. In a Bayesian approach, this probability is computed by integrating out all model parameters:

$$P(\mathbf{Z}|\mathbf{X}) = \int P(\mathbf{Z}, \phi|\mathbf{X})d\phi = \int P(\mathbf{Z}|\mathbf{X}, \phi)P(\phi|\mathbf{X})d\phi$$

Our HDP extension is depicted graphically in Figure 2(a). Similar to the HDP model, the distribution over events associated with each document, $\boldsymbol{\beta}$, is generated by a Dirichlet process with a concentration parameter $\alpha > 0$. Since this setting enables a clustering of event mentions at the document level, it is desirable that events be shared across documents and the number of events K be inferred from data. To ensure this flexibility, a global nonparametric DP prior with a hyperparameter γ and a global base measure H can be considered for $\boldsymbol{\beta}$ (Teh et al., 2006). The global distribution drawn from this DP prior, denoted as $\boldsymbol{\beta}_0$ in Figure 2(a), encodes the event mixing weights. Thus, same global events are used for each document, but each event has a document specific distribution β_i that is drawn from a DP prior centered on the global weights $\boldsymbol{\beta}_0$.

To infer the true posterior probability of $P(\mathbf{Z}|\mathbf{X})$, we follow (Teh et al., 2006) and use the *Gibbs sampling algorithm* (Geman and Geman, 1984) based on the direct assignment sampling scheme. In this sampling scheme, the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$ are integrated out analytically. Moreover, to reduce the complexity of computing $P(\mathbf{Z}|\mathbf{X})$, we make the naïve Bayes assumption that the feature variables $\mathbf{X}$ are conditionally independent given $\mathbf{Z}$. This allows us to factorize the joint distribution of feature variables $\mathbf{X}$ conditioned on $\mathbf{Z}$ into product of marginals. Thus, by Bayes rule, the formula for sampling an event index for mention j from document i, $Z_{i,j}$, is:[3]

$$P(Z_{i,j} \mid \mathbf{Z}^{-i,j}, \mathbf{X}) \propto P(Z_{i,j} \mid \mathbf{Z}^{-i,j}) \prod_{X \in \mathbf{X}} P(X_{i,j} | \mathbf{Z}, \mathbf{X}^{-i,j})$$

where $X_{i,j}$ represents the feature value of a feature type corresponding to the event mention j from the document i.

In the process of generating an event mention, an event index z is first sampled by using a mechanism that facilitates sampling from a prior for infinite mixture models called the *Chinese restaurant franchise* (CRF) representation, as reported in (Teh et al., 2006):

$$P(Z_{i,j} = z \mid \mathbf{Z}^{-i,j}, \beta_0) \propto \begin{cases} \alpha\beta_0^u, & \text{if } z = z_{new} \\ n_z + \alpha\beta_0^z, & \text{otherwise} \end{cases}$$

Here, n_z is the number of event mentions with event index z, z_{new} is a new event index not used already in $\mathbf{Z}^{-i,j}$, β_0^z are the global mixing proportions associated with the K events, and β_0^u is the weight for the unknown mixture component.

Next, to generate a feature value x (with the feature type X) of the event mention, the event z is

[2] In this subsection, the feature term is used in context of a feature type.

[3] $\mathbf{Z}^{-i,j}$ represents a notation for $\mathbf{Z} - \{Z_{i,j}\}$.

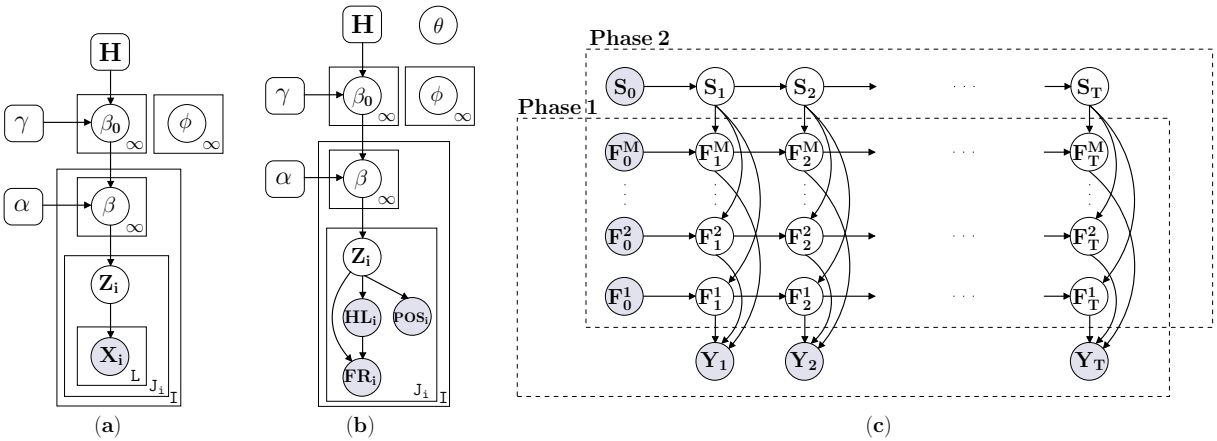


Figure 2: Graphical representation of our models: nodes correspond to random variables; shaded nodes denote observable variables; a rectangle captures the replication of the structure it contains, where the number of replications is indicated in the bottom-right corner. The model in (a) illustrates a flat representation of a limited number of features in a generalized framework (henceforth, HDP$_{flat}$). The model in (b) captures a simple example of structured network topology of three feature variables (henceforth, HDP$_{struct}$). The dependencies involving parameters ϕ and θ in these models are omitted for clarity. The model from (c) shows the representation of the iFHMM-iHMM model as well as the main phases of its generative process.

associated with a multinomial emission distribution over the feature values of X having the parameters $\boldsymbol{\phi} = \langle \phi^x_Z \rangle$. We assume that this emission distribution is drawn from a symmetric Dirichlet distribution with concentration λ_X:

$$P(X_{i,j} = x \mid \mathbf{Z}, \mathbf{X}^{-i,j}) \propto n_{x,z} + \lambda_X$$

where $X_{i,j}$ is the feature type of the mention j from the document i, and $n_{x,z}$ is the number of times the feature value x has been associated with the event index z in $(\mathbf{Z}, \mathbf{X}^{-i,j})$. We also apply the Lidstone's smoothing method to this distribution. In cases when only a feature type is considered (e.g., $\mathbf{X} = \langle HL \rangle$), the HDP$_{flat}$ model is identical with the original HDP model. We denote this one feature model by HDP$_{1f}$.

When dependencies between feature variables exist (e.g., in our case, frame elements are dependent on the semantic frames that define them, and frames are dependent on the words that evoke them), various global distributions are involved for computing $P(\mathbf{Z}|\mathbf{X})$. For the model depicted in Figure 2(b), for instance, the posterior probability is given by:

$$P(Z_{i,j}) P(FR_{i,j} \,|\, HL_{i,j}, \boldsymbol{\theta}) \prod_{X \in \mathbf{X}} P(X_{i,j} \,|\, \mathbf{Z})$$

In this formula, $P(FR_{i,j}|HL_{i,j}, \boldsymbol{\theta})$ is a global distribution parameterized by $\boldsymbol{\theta}$, and X is a feature variable from the set $\mathbf{X} = \langle HL, POS, FR \rangle$. For the sake of clarity, we omit the conditioning components of $\mathbf{Z}$, $\mathbf{HL}$, $\mathbf{FR}$, and $\mathbf{POS}$.

3.2 An Infinite Feature Model

To relax some of the restrictions of the first model, we devise an approach that combines the *infinite factorial hidden Markov model* (iFHMM) with the *infinite hidden Markov model* (iHMM) to form the iFHMM-iHMM model.

The iFHMM framework uses the *Markov Indian buffet process* (mIBP) (Van Gael et al., 2008b) in order to represent each object as a sparse subset of a potentially unbounded set of latent features (Griffiths and Ghahramani, 2006; Ghahramani et al., 2007; Van Gael et al., 2008a).[4] Specifically, the mIBP defines a distribution over an unbounded set of binary Markov chains, where each chain can be associated with a binary latent feature that evolves over time according to Markov dynamics. Therefore, if we denote by M the total number of feature chains and by T the number of observable components, the mIBP defines a probability distribution over a binary matrix $\mathbf{F}$ with T rows, which correspond to observations, and an unbounded number of columns M, which correspond to features. An observation y_t contains a subset from the unbounded set of features $\{f^1, f^2, \ldots, f^M\}$ that is represented in the matrix by a binary vector $\mathbf{F_t} = \langle F_t^1, F_t^2, \ldots, F_t^M \rangle$, where $F_t^i = 1$ indicates that f^i is associated with y_t. In other words, $\mathbf{F}$ decomposes the observations and represents them as feature factors, which can then be associated with hidden variables in an iFHMM model as depicted in Figure 2(c).

[4] In this subsection, a feature will be represented by a (feature type:feature value) pair.

Although the iFHMM allows a more flexible representation of the latent structure by letting the number of parallel Markov chains M be learned from data, it cannot be used as a framework where the number of clustering components K is infinite. On the other hand, the iHMM represents a nonparametric extension of the *hidden Markov model* (HMM) (Rabiner, 1989) that allows performing inference on an infinite number of states K. To further increase the representational power for modeling discrete time series data, we propose a nonparametric extension that combines the best of the two models, and lets the parameters M and K be learned from data.

As shown in Figure 2(c), each step in the new iHMM-iFHMM generative process is performed in two phases: (i) the latent feature variables from the iFHMM framework are sampled using the mIBP mechanism; and (ii) the features sampled so far, which become observable during this second phase, are used in an adapted version of the *beam sampling algorithm* (Van Gael et al., 2008a) to infer the clustering components (i.e., latent events).

In the first phase, the stochastic process for sampling features in $\mathbf{F}$ is defined as follows. The first component samples a number of $\mathsf{Poisson}(\alpha')$ features. In general, depending on the value that was sampled in the previous step $(t-1)$, a feature f^m is sampled for the t^{th} component according to the $P(F_t^m = 1 \mid F_{t-1}^m = 1)$ and $P(F_t^m = 1 \mid F_{t-1}^m = 0)$ probabilities.[5] After all features are sampled for the t^{th} component, a number of $\mathsf{Poisson}(\alpha'/t)$ new features are assigned for this component, and M gets incremented accordingly.

To describe the adapted beam sampler, which is employed in the second phase of the generative process, we introduce additional notations. We denote by $(s_1, \ldots, s_T)$ the sequence of hidden states corresponding to the sequence of event mentions $(y_1, \ldots, y_T)$, where each state s_t belongs to one of the K events, $s_t \in \{1, \ldots, K\}$, and each mention y_t is represented by a sequence of latent features $\langle F_t^1, F_t^2, \ldots, F_t^M \rangle$. One element of the transition probability $\boldsymbol{\pi}$ is defined as $\pi_{ij} = P(s_t = j \mid s_{t-1} = i)$, and a mention y_t is generated according to a likelihood model $\mathcal{F}$ that is parameterized by a state-dependent parameter ϕ_{s_t} $(y_t \mid s_t \sim \mathcal{F}(\phi_{s_t}))$. The observation parameters $\boldsymbol{\phi}$ are drawn independently from an identical prior base distribution H.

The beam sampling algorithm combines the ideas of slice sampling and dynamic programming for an efficient sampling of state trajectories. Since in time series models the transition probabilities have independent priors (Beal et al., 2002), Van Gael and colleagues (2008a) also used the HDP mechanism to allow couplings across transitions. For sampling the whole hidden state trajectory $\mathbf{s}$, this algorithm employs a forward filtering-backward sampling technique.

In the forward step of our adapted beam sampler, for each mention y_t, we sample features using the mIBP mechanism and the auxiliary variable $u_t \sim \mathsf{Uniform}(0, \pi_{s_{t-1}s_t})$. As explained in (Van Gael et al., 2008a), the auxiliary variables $\mathbf{u}$ are used to filter only those trajectories $\mathbf{s}$ for which $\pi_{s_{t-1}s_t} \geq u_t$ for all t. Also, in this step, we compute the probabilities $P(s_t \mid y_{1:t}, u_{1:t})$ for all t:

$$P(s_t|y_{1:t},u_{1:t}) \propto P(y_t|s_t) \sum_{s_{t-1}:u_t<\pi_{s_{t-1}s_t}} P(s_{t-1}|y_{1:t-1},u_{1:t-1})$$

Here, the dependencies involving parameters $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ are omitted for clarity.

In the backward step, we first sample the event for the last state s_T directly from $P(s_T \mid y_{1:T}, u_{1:T})$ and then, for all $t : T-1 \ldots 1$, we sample each state s_t given s_{t+1} by using the formula $P(s_t \mid s_{t+1}, y_{1:T}, u_{1:T}) \propto P(s_t \mid y_{1:t}, u_{1:t}) P(s_{t+1} \mid s_t, u_{t+1})$. To sample the emission distribution $\boldsymbol{\phi}$ efficiently, and to ensure that each mention is characterized by a finite set of representative features, we set the base distribution H to be conjugate with the data distribution $\mathcal{F}$ in a Dirichlet-multinomial model with the multinomial parameters $(o_1, \ldots, o_K)$ defined as:

$$o_k = \sum_{t=1}^{T} \sum_{f^m \in B_t} n_{mk}$$

In this formula, n_{mk} counts how many times the feature f^m was sampled for the event k, and B_t stores a finite set of features for y_t.

The mechanism for building a finite set of representative features for the mention y_t is based on *slice sampling* (Neal, 2003). Letting q_m be the number of times the feature f^m was sampled in the mIBP, and v_t an auxiliary variable for y_t such that $v_t \sim \mathsf{Uniform}(1, \max\{q_m : F_t^m = 1\})$, we define the finite feature set B_t for the observation y_t as $B_t = \{f^m : F_t^m = 1 \wedge q_m \geq v_t\}$. The finiteness of this feature set is based on the observation that, in the generative process of the mIBP, only a finite set

[5] Technical details for computing these probabilities are described in (Van Gael et al., 2008b).

of features are sampled for a component. We denote this model as iFHMM-iHMM$_{uniform}$. Also, it is worth mentioning that, by using this type of sampling, only the most representative features of y_t get selected in B_t.

Furthermore, we explore the mechanism for selecting a finite set of features associated with an observation by: (1) considering all the observation's features whose corresponding feature counter $q_m \geq 1$ ($unfiltered$); (2) selecting only the higher half of the feature distribution consisting of the observation's features that were sampled at least once in the mIBP model ($median$); and (3) sampling v_t from a discrete distribution of the observation's features that were sampled at least once in the mIBP ($discrete$).

4 Experiments

Datasets One dataset we employed is the automatic content extraction (ACE) (ACE-Event, 2005). However, the utilization of the ACE corpus for the task of solving event coreference is limited because this resource provides only within-document event coreference annotations using a restricted set of event types such as LIFE, BUSINESS, CONFLICT, and JUSTICE. Therefore, as a second dataset, we created the **EventCorefBank** (ECB) corpus[6] to increase the diversity of event types and to be able to evaluate our models for both within- and cross-document event coreference resolution. One important step in the creation process of this corpus consists in finding sets of related documents that describe the same seminal event such that the annotation of coreferential event mentions across documents is possible. For this purpose, we selected from the GoogleNews archive[7] various topics whose description contains keywords such as *commercial transaction*, *attack*, *death*, *sports*, *terrorist act*, *election*, *arrest*, *natural disaster*, etc. The entire annotation process for creating the ECB resource is described in (Bejan and Harabagiu, 2008). Table 1 lists several basic statistics extracted from these two corpora.

Evaluation For a more realistic approach, we not only trained the models on the manually annotated event mentions (i.e., true mentions), but also on all the possible mentions encoded in the two datasets. To extract all event mentions, we ran the event identifier described in (Bejan, 2007). The mentions extracted by this system (i.e., system mentions) were able to cover all the true mentions from both datasets. As shown in Table 1, we extracted from ACE and ECB corpora 45289 and 21175 system mentions, respectively.

	ACE	ECB
Number of topics	–	43
Number of documents	745	482
Number of within-topic events	–	339
Number of cross-document events	–	208
Number of within-document events	4946	1302
Number of true mentions	6553	1744
Number of system mentions	45289	21175
Number of distinct feature values	391798	237197

Table 1: Statistics of the ACE and ECB corpora.

We report results in terms of recall (R), precision (P), and F-score (F) by employing the *mention*-based B^3 metric (Bagga and Baldwin, 1998), the *entity*-based CEAF metric (Luo, 2005), and the pairwise F1 (PW) metric. All the results are averaged over 5 runs of the generative models. In the evaluation process, we considered only the true mentions of the ACE test dataset, and the event mentions of the test sets derived from a 5-fold cross validation scheme on the ECB dataset. For evaluating the cross-document coreference annotations, we adopted the same approach as described in (Bagga and Baldwin, 1999) by merging all the documents from the same topic into a meta-document and then scoring this document as performed for within-document evaluation. For both corpora, we considered a set of 132 feature types, where each feature type consists on average of 3900 distinct feature values.

Baselines We consider two baselines for event coreference resolution (rows 1&2 in Tables 2&3). One baseline groups each event mention by its event class (BL$_{eclass}$). Therefore, for this baseline, we cluster mentions according to their corresponding EC feature value. Similarly, the second baseline uses as grouping criteria for event mentions their corresponding WNS feature value (BL$_{syn}$).

HDP Extensions Due to memory limitations, we evaluated the HDP models on a restricted set of manually selected feature types. In general, the HDP$_{1f}$ model with the feature type HL, which plays the role of a baseline for the HDP$_{flat}$ and HDP$_{struct}$ models, outperforms both baselines on the ACE and ECB datasets. For the HDP$_{flat}$ models (rows 4–7 in Tables 2&3), we classified the experiments according to the set of feature types described in Section 2. Our experiments reveal that the best configuration of features for this model

[6] ECB is available at http://www.hlt.utdallas.edu/∼ady.
[7] http://news.google.com/

	Model configuration	B^3			CEAF			PW			B^3			CEAF			PW		
		R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
		ECB | WD									**ECB | CD**								
1	BL_{eclass}	97.7	55.8	71.0	44.5	80.1	57.2	93.7	25.4	39.8	93.8	49.6	64.9	36.6	72.7	48.7	90.7	28.6	43.3
2	BL_{syn}	91.5	57.4	70.5	45.7	75.9	57.0	65.3	21.9	32.6	84.6	48.1	61.3	32.8	63.6	43.3	66.2	26.0	37.3
3	HDP_{1f} (HL)	84.3	89.0	86.5	83.4	79.6	81.4	36.6	53.4	42.6	67.0	86.2	75.3	76.2	57.1	65.2	34.9	58.9	43.5
4	HDP_{flat} (LF)	81.4	98.2	89.0	92.7	77.2	84.2	24.7	82.8	37.7	63.8	97.3	77.0	84.9	54.3	66.1	27.2	88.5	41.5
5	(LF+CF)	81.5	98.0	89.0	92.8	77.9	84.7	24.6	80.7	37.4	64.6	97.3	77.6	85.3	55.6	67.2	27.6	88.7	42.0
6	(LF+CF+WF)	82.0	98.9	89.6	93.7	78.4	85.3	26.8	89.9	41.0	65.8	98.0	78.7	86.7	57.1	68.8	29.6	93.0	44.8
7	(LF+CF+WF+SF)	82.1	99.2	89.8	93.9	78.2	85.3	27.0	92.4	41.3	65.0	98.7	78.3	86.9	56.0	68.0	29.2	95.1	44.4
8	HDP_{struct} (HL→FR→FEA)	84.3	97.1	**90.2**	92.7	81.1	**86.5**	34.4	83.0	**48.6**	69.3	95.8	**80.4**	86.2	60.1	**70.8**	37.5	85.6	**52.1**
9	iFHMM-iHMM$_{unfiltered}$	82.6	97.7	89.5	92.7	78.5	85.0	28.5	82.4	41.8	67.2	96.4	79.1	85.6	58.0	69.1	32.5	87.7	47.2
10	iFHMM-iHMM$_{discrete}$	82.6	98.1	89.7	93.2	79.0	85.5	29.7	85.4	44.0	66.2	96.2	78.4	84.8	57.2	68.3	32.2	88.1	47.1
11	iFHMM-iHMM$_{median}$	82.6	97.8	89.5	92.9	78.8	85.3	29.3	83.7	43.0	67.0	96.5	79.0	86.1	58.3	69.5	33.1	88.1	47.9
12	iFHMM-iHMM$_{uniform}$	82.5	98.1	89.6	93.1	78.8	85.3	29.4	86.6	43.7	67.0	96.4	79.0	85.5	58.0	69.1	33.3	88.3	48.2

Table 2: Results for within-document (WD) and cross-document (WD) coreference resolution on the ECB dataset.

	B^3			CEAF			PW		
	R	P	F	R	P	F	R	P	F
	ACE | WD								
1	97.9	25.0	39.9	14.7	64.4	24.0	93.5	8.2	15.2
2	89.3	36.7	52.1	25.1	64.8	36.2	63.8	10.5	18.1
3	86.0	70.6	77.5	62.3	76.4	68.6	50.5	27.7	35.8
4	82.9	82.6	82.7	74.9	75.8	75.3	42.4	41.9	42.1
5	82.0	84.9	83.4	77.8	75.3	76.6	37.9	45.1	41.2
6	83.3	83.6	83.4	76.3	76.2	76.3	42.2	43.9	43.0
7	83.4	84.2	**83.8**	76.9	76.5	**76.7**	43.3	47.1	**45.1**
8	86.2	76.9	81.3	69.0	77.5	73.0	53.2	38.1	44.4
9	82.8	83.6	83.2	75.8	75.0	75.4	41.4	42.6	42.0
10	83.1	81.5	82.3	73.7	75.1	74.4	41.9	40.1	41.0
11	83.0	81.3	82.1	73.2	75.2	74.2	40.7	39.0	39.8
12	81.9	82.2	82.1	74.6	74.5	74.5	37.2	39.0	38.1

Table 3: Results for WD coreference resolution on ACE.

consists of a combination of feature types from all the categories of features (row 7). For the HDP$_{struct}$ experiments, we considered the set of features of the best HDP$_{flat}$ experiment as well as the dependencies between HL, FR, and FEA. Overall, we can assert that HDP$_{flat}$ achieved the best performance results on the ACE test dataset (Table 3), whereas HDP$_{struct}$ proved to be more effective on the ECB dataset (Table 2). Moreover, the results of the HDP$_{flat}$ and HDP$_{struct}$ models show an F-score increase by 4-10% over HDP$_{1f}$, and therefore, the results prove that the HDP extension provides a more flexible representation for clustering objects with rich properties.

We also plot the evolution of our generative processes. For instance, Figure 3(a) shows that the HDP$_{flat}$ model corresponding to row 7 in Table 3 converges in 350 iteration steps to a posterior distribution over event mentions from ACE with around 2000 latent events. Additionally, our experiments with different values of the λ parameter for the Lidstone's smoothing method indicate that this smoothing method is useful for improving the performance of the HDP models. However, we could not find a λ value in our experiments that brings a major improvement over the non-smoothed HDP models. Figure3(b) shows the performances of HDP$_{struct}$ on ECB with various λ values.[8] The HDP results from Tables 2&3 correspond to a λ value of 10^{-4} and 10^{-2} for HDP$_{flat}$ and HDP$_{struct}$, respectively.

iFHMM-iHMM In spite of the fact that the iFHMM-iHMM model employs automatic feature selection, its results remain competitive against the results of the HDP models, where the feature types were manually tuned. When comparing the strategies for filtering feature values in this framework, we could not find a distinct separation between the results obtained by the $unfiltered$, $discrete$, $median$, and $uniform$ models. As observed from Tables 2&3, most of the iFHMM-iHMM results fall in between the HDP$_{flat}$ and HDP$_{struct}$ results. The results were obtained by automatically selecting only up to 1.5% of distinct feature values. Figure 3(c) shows the percents of features employed by this model for various values of the parameter α' that controls the number of sampled features. The best results (also listed in Tables 2&3) were obtained for $\alpha' = 10$ (0.05%) on ACE and $\alpha' = 150$ (0.91%) on ECB.

To show the usefulness of the sampling schemes considered for this model, we also compare in Table 4 the results obtained by an iFHMM-iHMM model that considers all the feature values associated with an observable object (iFHMM-iHMM$_{all}$) against the iFHMM-iHMM models that employ the mIBP sampling scheme together with the $unfiltered$, $discrete$, $median$, and $uniform$ filtering schemes. Because of the memory limitation constraints, we performed the experiments listed in Table 4 by selecting only a subset from

[8] A configuration $\lambda = 0$ in the Lidstone's smoothing method is equivalent with a non-smoothed version of the model on which it is applied.

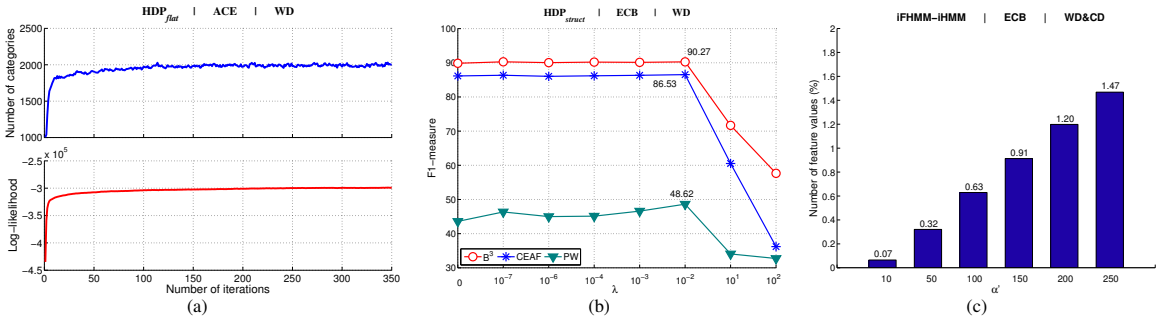


Figure 3: (a) Evolution of K and log-likelihood in the HDP$_{flat}$ model. (b) Evaluation of the Lidstone's smoothing method in the HDP$_{struct}$ model. (c) Counts of features employed by the iFHMM-iHMM model for various α' values.

Model	B^3			CEAF			PW		
	R	P	F	R	P	F	R	P	F
ACE | WD									
all	89.3	39.8	55.0	30.2	68.8	42.0	62.7	9.1	15.9
unfiltered	83.3	77.7	80.4	70.6	75.9	73.2	42.1	34.6	38.0
discrete	83.8	80.7	82.2	73.0	75.8	74.4	43.9	39.1	41.4
median	83.5	80.2	81.8	72.2	75.3	73.7	42.7	38.2	40.3
uniform	82.8	80.7	81.7	72.8	75.2	73.9	41.4	39.3	40.3
ECB | WD									
all	89.5	62.5	73.6	53.3	76.5	62.8	60.7	22.9	33.2
unfiltered	82.6	96.6	89.0	92.0	79.1	85.1	28.4	75.6	41.0
discrete	83.1	96.7	89.4	91.6	79.2	84.9	30.5	79.0	43.9
median	82.5	97.3	89.3	92.8	78.9	85.3	29.2	78.8	42.0
uniform	82.7	96.0	88.9	91.1	79.0	84.6	29.3	74.9	41.6
ECB | CD									
all	79.3	54.4	64.5	43.3	61.3	50.7	59.6	26.2	36.4
unfiltered	67.2	94.5	78.5	84.7	59.2	69.6	32.8	82.5	46.8
discrete	67.6	94.8	78.9	83.8	58.3	68.8	34.3	85.3	48.9
median	66.7	95.2	78.4	84.5	57.7	68.5	32.2	83.7	46.3
uniform	67.7	93.6	78.4	83.6	59.2	69.2	33.6	79.5	46.9

Table 4: Feature non-sampling vs. feature sampling in the iFHMM-iHMM model.

the feature types which proved to be salient in the HDP experiments. As listed in Table 4, all the iFHMM-iHMM models that used a feature sampling scheme significantly outperform the iFHMM-iHMM$_{all}$ model; this proves that all the sampling schemes considered in the iFHMM-iHMM framework are able to successfully filter out noisy and redundant feature values.

The closest comparison to prior work is the supervised approach described in (Chen and Ji, 2009) that achieved a 92.2% B^3 F-measure on the ACE corpus. However, for this result, ground truth event mentions as well as a manually tuned coreference threshold were employed.

5 Error Analysis

One frequent error occurs when a more complex form of semantic inference is needed to find a correspondence between two event mentions of the same individuated event. For instance, since all properties and participants of em_3(*deal*) are omitted in our example and no common features exist between em_3(*buy*) and em_1(*buy*) to indicate a similarity between these mentions, they will most probably be assigned to different clusters. This example also suggests the need for a better modeling of the discourse salience for event mentions.

Another common error is made when matching the semantic roles corresponding to coreferential event mentions. Although we simulated entity coreference by using various semantic features, the task of matching participants of coreferential event mentions is not completely solved. This is because, in many coreferential cases, partonomic relations between semantic roles need to be inferred.[9] Examples of such relations extracted from ECB are *Israeli forces* $\xrightarrow{\text{PART OF}}$ *Israel*, *an Indian warship* $\xrightarrow{\text{PART OF}}$ *the Indian navy*, *his cell* $\xrightarrow{\text{PART OF}}$ *Sicilian jail*. Similarly for event properties, many coreferential examples do not specify a clear location and time interval (e.g., *Jabaliya refugee camp* $\xrightarrow{\text{PART OF}}$ *Gaza*, *Tuesday* $\xrightarrow{\text{PART OF}}$ *this week*). In future work, we plan to build relevant clusters using partonomies and taxonomies such as the WordNet hierarchies built from MERONYMY/HOLONYMY and HYPERNYMY/HYPONYMY relations, respectively.[10]

6 Conclusion

We have presented two novel, nonparametric Bayesian models that are designed to solve complex problems that require clustering objects characterized by a rich set of properties. Our experiments for event coreference resolution proved that these models are able to solve real data applications in which the feature and cluster numbers are treated as free parameters, and the selection of feature values is performed automatically.

[9] This observation was also reported in (Hasler and Orasan, 2009). [10] This task is not trivial since, if applying the transitive closure on these relations, all words will end up being part from the same cluster with *entity* for instance.

References

ACE-Event. 2005. ACE (Automatic Content Extraction) English Annotation Guidelines for Events, version 5.4.3 2005.07.01.

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic Detection and Tracking Pilot Study: Final Report. In *Proceedings of the Broadcast News Understanding and Transcription Workshop*, pages 194–218.

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC-1998).*

Amit Bagga and Breck Baldwin. 1999. Cross-Document Event Coreference: Annotations, Experiments, and Observations. In *Proceedings of the ACL Workshop on Coreference and its Applications*, pages 1–8.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL).*

Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. 2002. The Infinite Hidden Markov Model. In *Advances in Neural Information Processing Systems 14 (NIPS).*

Cosmin Adrian Bejan and Sanda Harabagiu. 2008. A Linguistic Resource for Discovering Event Structures and Resolving Event Coreference. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC).*

Cosmin Adrian Bejan and Chris Hathaway. 2007. UTD-SRL: A Pipeline Architecture for Extracting Frame Semantic Structures. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval)*, pages 460–463.

Cosmin Adrian Bejan, Matthew Titsworth, Andrew Hickl, and Sanda Harabagiu. 2009. Nonparametric Bayesian Models for Unsupervised Event Coreference Resolution. In *Advances in Neural Information Processing Systems 23 (NIPS).*

Cosmin Adrian Bejan. 2007. Deriving Chronological Information from Texts through a Graph-based Algorithm. In *Proceedings of the 20th Florida Artificial Intelligence Research Society International Conference (FLAIRS), Applied Natural Language Processing track.*

Zheng Chen and Heng Ji. 2009. Graph-based Event Coreference Resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*, pages 54–57.

Donald Davidson, 1969. *The Individuation of Events*. In N. Rescher et al., eds., *Essays in Honor of Carl G. Hempel*, Dordrecht: Reidel. Reprinted in D. Davidson, ed., *Essays on Actions and Events*, 2001, Oxford: Clarendon Press.

Donald Davidson, 1985. *Reply to Quine on Events*, pages 172–176. In E. LePore and B. McLaughlin, eds., *Actions and Events: Perspectives on the Philosophy of Donald Davidson*, Oxford: Blackwell.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding Contradictions in Text. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 1039–1047.

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

Thomas S. Ferguson. 1973. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230.

Charles J. Fillmore. 1982. Frame Semantics. In *Linguistics in the Morning Calm*.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Zoubin Ghahramani, T. L. Griffiths, and Peter Sollich, 2007. *Bayesian Statistics 8*, chapter Bayesian nonparametric latent feature models, pages 201–225. Oxford University Press.

Tom Griffiths and Zoubin Ghahramani. 2006. Infinite Latent Feature Models and the Indian Buffet Process. In *Advances in Neural Information Processing Systems 18 (NIPS)*, pages 475–482.

Aria Haghighi and Dan Klein. 2007. Unsupervised Coreference Resolution in a Nonparametric Bayesian Model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 848–855.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust Textual Inference via Graph Matching. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 387–394.

Laura Hasler and Constantin Orasan. 2009. Do coreferential arguments make event mentions coreferential? In *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2009).*

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of the Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts, 35th Meeting of ACL*, pages 75–81.

John B. Lowe, Collin F. Baker, and Charles J. Fillmore. 1997. A frame-semantic approach to semantic annotation. In *Proceedings of the SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 18–24.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (EMNLP-2005)*, pages 25–32.

Jeff Malpas. 2009. Donald Davidson. In *The Stanford Encyclopedia of Philosophy (Fall 2009 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/fall2009/entries/davidson/*.

Srini Narayanan and Sanda Harabagiu. 2004. Question Answering Based on Semantic Structures. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 693–701.

Radford M. Neal. 2003. Slice Sampling. *The Annals of Statistics*, 31:705–741.

Vincent Ng. 2008. Unsupervised Models for Coreference Resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 640–649.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–105.

Hoifung Poon and Pedro Domingos. 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–659.

James Pustejovsky, Jose Castano, Bob Ingria, Roser Sauri, Rob Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS)*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The TimeBank Corpus. In *Corpus Linguistics*, pages 647–656.

W. V. O. Quine, 1985. *Events and Reification*, pages 162–171. In E. LePore and B. P. McLaughlin, eds., *Actions and Events: Perspectives on the philosophy of Donald Davidson*, Oxford: Blackwell. Reprinted in R. Casati and A. C. Varzi, eds., *Events*, 1996, pages 107–116, Aldershot: Dartmouth.

Lawrence R. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, pages 257–286.

Yee Whye Teh, Michael Jordan, Matthew Beal, and David Blei. 2006. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Jurgen Van Gael, Y. Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008a. Beam Sampling for the Infinite Hidden Markov Model. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML)*, pages 1088–1095.

Jurgen Van Gael, Yee Whye Teh, and Zoubin Ghahramani. 2008b. The Infinite Factorial Hidden Markov Model. In *Advances in Neural Information Processing Systems 21 (NIPS)*.

Coreference Resolution across Corpora: Languages, Coding Schemes, and Preprocessing Information

Marta Recasens
CLiC - University of Barcelona
Gran Via 585
Barcelona, Spain
mrecasens@ub.edu

Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey CA, USA
hovy@isi.edu

Abstract

This paper explores the effect that different corpus configurations have on the performance of a coreference resolution system, as measured by MUC, B^3, and CEAF. By varying separately three parameters (language, annotation scheme, and preprocessing information) and applying the same coreference resolution system, the strong bonds between system and corpus are demonstrated. The experiments reveal problems in coreference resolution evaluation relating to task definition, coding schemes, and features. They also expose systematic biases in the coreference evaluation metrics. We show that system comparison is only possible when corpus parameters are in exact agreement.

1 Introduction

The task of coreference resolution, which aims to automatically identify the expressions in a text that refer to the same discourse entity, has been an increasing research topic in NLP ever since MUC-6 made available the first coreferentially annotated corpus in 1995. Most research has centered around the rules by which mentions are allowed to corefer, the features characterizing mention pairs, the algorithms for building coreference chains, and coreference evaluation methods. The surprisingly important role played by different aspects of the corpus, however, is an issue to which little attention has been paid. We demonstrate the extent to which a system will be evaluated as performing differently depending on parameters such as the corpus language, the way coreference relations are defined in the corresponding coding scheme, and the nature and source of preprocessing information.

This paper unpacks these issues by running the same system—a prototype entity-based architecture called CISTELL—on different corpus configurations, varying three parameters. First, we show how much language-specific issues affect performance when trained and tested on English and Spanish. Second, we demonstrate the extent to which the specific annotation scheme (used on the same corpus) makes evaluated performance vary. Third, we compare the performance using gold-standard preprocessing information with that using automatic preprocessing tools.

Throughout, we apply the three principal coreference evaluation measures in use today: MUC, B^3, and CEAF. We highlight the systematic preferences of each measure to reward different configurations. This raises the difficult question of why one should use one or another evaluation measure, and how one should interpret their differences in reporting changes of performance score due to 'secondary' factors like preprocessing information.

To this end, we employ three corpora: ACE (Doddington et al., 2004), OntoNotes (Pradhan et al., 2007), and AnCora (Recasens and Martí, 2009). In order to isolate the three parameters as far as possible, we benefit from a 100k-word portion (from the TDT collection) that is common to both ACE and OntoNotes. We apply the same coreference resolution system in all cases. The results show that a system's score is not informative by itself, as different corpora or corpus parameters lead to different scores. Our goal is not to achieve the best performance to date, but rather to expose various issues raised by the choices of corpus preparation and evaluation measure and to shed light on the definition, methods, evaluation, and complexities of the coreference resolution task.

The paper is organized as follows. Section 2 sets our work in context and provides the motivations for undertaking this study. Section 3 presents the architecture of CISTELL, the system used in the experimental evaluation. In Sections 4, 5,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1423–1432,
Uppsala, Sweden, 11-16 July 2010.

and 6, we describe the experiments on three different datasets and discuss the results. We conclude in Section 7.

2 Background

The bulk of research on automatic coreference resolution to date has been done for English and used two different types of corpus: MUC (Hirschman and Chinchor, 1997) and ACE (Doddington et al., 2004). A variety of learning-based systems have been trained and tested on the former (Soon et al., 2001; Uryupina, 2006), on the latter (Culotta et al., 2007; Bengtson and Roth, 2008; Denis and Baldridge, 2009), or on both (Finkel and Manning, 2008; Haghighi and Klein, 2009). Testing on both is needed given that the two annotation schemes differ in some aspects. For example, only ACE includes singletons (mentions that do not corefer) and ACE is restricted to seven semantic types.[1] Also, despite a critical discussion in the MUC task definition (van Deemter and Kibble, 2000), the ACE scheme continues to treat nominal predicates and appositive phrases as coreferential.

A third coreferentially annotated corpus—the largest for English—is OntoNotes (Pradhan et al., 2007; Hovy et al., 2006). Unlike ACE, it is not application-oriented, so coreference relations between all types of NPs are annotated. The identity relation is kept apart from the attributive relation, and it also contains gold-standard morphological, syntactic and semantic information.

Since the MUC and ACE corpora are annotated with only coreference information,[2] existing systems first preprocess the data using automatic tools (POS taggers, parsers, etc.) to obtain the information needed for coreference resolution. However, given that the output from automatic tools is far from perfect, it is hard to determine the level of performance of a coreference module acting on gold-standard preprocessing information. OntoNotes makes it possible to separate the coreference resolution problem from other tasks.

Our study adds to the previously reported evidence by Stoyanov et al. (2009) that differences in corpora and in the task definitions need to be taken into account when comparing coreference resolution systems. We provide new insights as the current analysis differs in four ways. First, Stoyanov et al. (2009) report on differences between MUC and ACE, while we contrast ACE and OntoNotes. Given that ACE and OntoNotes include some of the same texts but annotated according to their respective guidelines, we can better isolate the effect of differences as well as add the additional dimension of gold preprocessing. Second, we evaluate not only with the MUC and B^3 scoring metrics, but also with CEAF. Third, all our experiments use true mentions[3] to avoid effects due to spurious system mentions. Finally, including different baselines and variations of the resolution model allows us to reveal biases of the metrics.

Coreference resolution systems have been tested on languages other than English only within the ACE program (Luo and Zitouni, 2005), probably due to the fact that coreferentially annotated corpora for other languages are scarce. Thus there has been no discussion of the extent to which systems are portable across languages. This paper studies the case of English and Spanish.[4]

Several coreference systems have been developed in the past (Culotta et al., 2007; Finkel and Manning, 2008; Poon and Domingos, 2008; Haghighi and Klein, 2009; Ng, 2009). It is not our aim to compete with them. Rather, we conduct three experiments under a specific setup for comparison purposes. To this end, we use a different, neutral, system, and a dataset that is small and different from official ACE test sets despite the fact that it prevents our results from being compared directly with other systems.

3 Experimental Setup

3.1 System Description

The system architecture used in our experiments, CISTELL, is based on the incrementality of discourse. As a discourse evolves, it constructs a model that is updated with the new information gradually provided. A key element in this model are the entities the discourse is about, as they form the discourse backbone, especially those that are mentioned multiple times. Most entities, however, are only mentioned once. Consider the growth of the entity *Mount Popocatépetl* in (1).[5]

[1] The ACE-2004/05 semantic types are person, organization, geo-political entity, location, facility, vehicle, weapon.

[2] ACE also specifies entity types and relations.

[3] The adjective *true* contrasts with *system* and refers to the gold standard.

[4] Multilinguality is one of the focuses of SemEval-2010 Task 1 (Recasens et al., 2010).

[5] Following the ACE terminology, we use the term *mention* for an instance of reference to an object, and *entity* for a collection of mentions referring to the same object. Entities

(1) We have an update tonight on [this, the volcano in Mexico, they call El Popo]$_{m3}$ ... As the sun rises over [Mt. Popo]$_{m7}$ tonight, the only hint of the fire storm inside, whiffs of smoke, but just a few hours earlier, [the volcano]$_{m11}$ exploding spewing rock and red-hot lava. [The fourth largest mountain in North America, nearly 18,000 feet high]$_{m15}$, erupting this week with [its]$_{m20}$ most violent outburst in 1,200 years.

Mentions can be pronouns ($m20$), they can be a (shortened) string repetition using either the name ($m7$) or the type ($m11$), or they can add new information about the entity: $m15$ provides the supertype and informs the reader about the height of the volcano and its ranking position.

In CISTELL,[6] discourse entities are conceived as 'baskets': they are empty at the beginning of the discourse, but keep growing as new attributes (e.g., name, type, location) are predicated about them. Baskets are filled with this information, which can appear within a mention or elsewhere in the sentence. The ever-growing amount of information in a basket allows richer comparisons to new mentions encountered in the text.

CISTELL follows the learning-based coreference architecture in which the task is split into classification and clustering (Soon et al., 2001; Bengtson and Roth, 2008) but combines them simultaneously. Clustering is identified with basket-growing, the core process, and a pairwise classifier is called every time CISTELL considers whether a basket must be clustered into a (growing) basket, which might contain one or more mentions. We use a memory-based learning classifier trained with TiMBL (Daelemans and Bosch, 2005). Basket-growing is done in four different ways, explained next.

3.2 Baselines and Models

In each experiment, we compute three baselines (1, 2, 3), and run CISTELL under four different models (4, 5, 6, 7).

1. ALL SINGLETONS. No coreference link is ever created. We include this baseline given the high number of singletons in the datasets, since some evaluation measures are affected by large numbers of singletons.
2. HEAD MATCH. All non-pronominal NPs that have the same head are clustered into the same entity.
3. HEAD MATCH + PRON. Like HEAD MATCH, plus allowing personal and possessive pronouns to link to the closest noun with which they agree in gender and number.
4. STRONG MATCH. Each mention (e.g., m_{11}) is paired with previous mentions starting from the beginning of the document (m_1–m_{11}, m_2–m_{11}, etc.).[7] When a pair (e.g., m_3–m_{11}) is classified as coreferent, additional pairwise checks are performed with all the mentions contained in the (growing) entity basket (e.g., m_7–m_{11}). Only if *all* the pairs are classified as coreferent is the mention under consideration attached to the existing growing entity. Otherwise, the search continues.[8]
5. SUPER STRONG MATCH. Similar to STRONG MATCH but with a threshold. Coreference pairwise classifications are only accepted when TiMBL distance is smaller than 0.09.[9]
6. BEST MATCH. Similar to STRONG MATCH but following Ng and Cardie (2002)'s best link approach. Thus, the mention under analysis is linked to the *most confident* mention among the previous ones, using TiMBL's confidence score.
7. WEAK MATCH. A simplified version of STRONG MATCH: not all mentions in the growing entity need to be classified as coreferent with the mention under analysis. A single positive pairwise decision suffices for the mention to be clustered into that entity.[10]

3.3 Features

We follow Soon et al. (2001), Ng and Cardie (2002) and Luo et al. (2004) to generate most of the 29 features we use for the pairwise model. These include features that capture information from different linguistic levels: textual strings (head match, substring match, distance, frequency), morphology (mention type, coordination, possessive phrase, gender match, number match), syntax (nominal predicate, apposition, relative clause, grammatical function), and semantic match (named-entity type, is-a type, supertype).

containing one single mention are referred to as *singletons*.

[6] 'Cistell' is the Catalan word for 'basket.'

[7] The opposite search direction was also tried but gave worse results.

[8] Taking the first mention classified as coreferent follows Soon et al. (2001)'s first-link approach.

[9] In TiMBL, being a memory-based learner, the closer the distance to an instance, the more confident the decision. We chose 0.09 because it appeared to offer the best results.

[10] STRONG and WEAK MATCH are similar to Luo et al. (2004)'s entity-mention and mention-pair models.

For Spanish, we use 34 features as a few variations are needed for language-specific issues such as zero subjects (Recasens and Hovy, 2009).

3.4 Evaluation

Since they sometimes provide quite different results, we evaluate using three coreference measures, as there is no agreement on a standard.

- MUC (Vilain et al., 1995). It computes the number of links common between the true and system partitions. Recall (R) and precision (P) result from dividing it by the minimum number of links required to specify the true and the system partitions, respectively.
- B^3 (Bagga and Baldwin, 1998). R and P are computed for each mention and averaged at the end. For each mention, the number of common mentions between the true and the system entity is divided by the number of mentions in the true entity or in the system entity to obtain R and P, respectively.
- CEAF (Luo, 2005). It finds the best one-to-one alignment between true and system entities. Using true mentions and the ϕ_3 similarity function, R and P are the same and correspond to the number of common mentions between the aligned entities divided by the total number of mentions.

4 Parameter 1: Language

The first experiment compared the performance of a coreference resolution system on a Germanic and a Romance language—English and Spanish—to explore to what extent language-specific issues such as zero subjects[11] or grammatical gender might influence a system.

Although OntoNotes and AnCora are two different corpora, they are very similar in those aspects that matter most for the study's purpose: they both include a substantial amount of texts belonging to the same genre (news) and manually annotated from the morphological to the semantic levels (POS tags, syntactic constituents, NEs, WordNet synsets, and coreference relations). More importantly, very similar coreference annotation guidelines make AnCora the ideal Spanish counterpart to OntoNotes.

[11]Most Romance languages are pro-drop allowing zero subject pronouns, which can be inferred from the verb.

Datasets Two datasets of similar size were selected from AnCora and OntoNotes in order to rule out corpus size as an explanation of any difference in performance. Corpus statistics about the distribution of mentions and entities are shown in Tables 1 and 2. Given that this paper is focused on coreference between NPs, the number of mentions only includes NPs. Both AnCora and OntoNotes annotate only multi-mention entities (i.e., those containing two or more coreferent mentions), so singleton entities are assumed to correspond to NPs with no coreference annotation.

Apart from a larger number of mentions in Spanish (Table 1), the two datasets look very similar in the distribution of singletons and multi-mention entities: about 85% and 15%, respectively. Multi-mention entities have an average of 3.9 mentions per entity in AnCora and 3.5 in OntoNotes. The distribution of mention types (Table 2), however, differs in two important respects: AnCora has a smaller number of personal pronouns as Spanish typically uses zero subjects, and it has a smaller number of bare NPs as the definite article accompanies more NPs than in English.

Results and Discussion Table 3 presents CISTELL's results for each dataset. They make evident problems with the evaluation metrics, namely the fact that the generated rankings are contradictory (Denis and Baldridge, 2009). They are consistent across the two corpora though: MUC rewards WEAK MATCH the most, B^3 rewards HEAD MATCH the most, and CEAF is divided between SUPER STRONG MATCH and BEST MATCH.

These preferences seem to reveal weaknesses of the scoring methods that make them biased towards a type of output. The model preferred by MUC is one that clusters many mentions together, thus getting a large number of correct coreference links (notice the high R for WEAK MATCH), but

	AnCora	OntoNotes
Pronouns	14.09	17.62
Personal pronouns	2.00	12.10
Zero subject pronouns	6.51	–
Possessive pronouns	3.57	2.96
Demonstrative pronouns	0.39	1.83
Definite NPs	37.69	20.67
Indefinite NPs	7.17	8.44
Demonstrative NPs	1.98	3.41
Bare NPs	33.02	42.92
Misc.	6.05	6.94

Table 2: Mention types (%) in Table 1 datasets.

		#docs	#words	#mentions	#entities (e)	#singleton e	#multi-mention e
AnCora	Training	955	299,014	91,904	64,535	54,991	9,544
	Test	30	9,851	2,991	2,189	1,877	312
OntoNotes	Training	850	301,311	74,692	55,819	48,199	7,620
	Test	33	9,763	2,463	1,790	1,476	314

Table 1: Corpus statistics for the large portion of OntoNotes and AnCora.

	MUC			B^3			CEAF
	P	R	F	P	R	F	P / R / F
AnCora - Spanish							
1. ALL SINGLETONS	–	–	–	100	73.32	84.61	73.32
2. HEAD MATCH	55.03	37.72	44.76	91.12	79.88	**85.13**	75.96
3. HEAD MATCH + PRON	48.22	44.24	46.14	86.21	80.66	83.34	76.30
4. STRONG MATCH	45.64	51.88	48.56	80.13	82.28	81.19	75.79
5. SUPER STRONG MATCH	45.68	36.47	40.56	86.10	79.09	82.45	**77.20**
6. BEST MATCH	43.10	35.59	38.98	85.24	79.67	82.36	75.23
7. WEAK MATCH	45.73	65.16	**53.75**	68.50	87.71	76.93	69.21
OntoNotes - English							
1. ALL SINGLETONS	–	–	–	100	72.68	84.18	72.68
2. HEAD MATCH	55.14	39.08	45.74	90.65	80.87	**85.48**	76.05
3. HEAD MATCH + PRON	47.10	53.05	49.90	82.28	83.13	82.70	75.15
4. STRONG MATCH	47.94	55.42	51.41	81.13	84.30	82.68	78.03
5. SUPER STRONG MATCH	48.27	47.55	47.90	84.00	82.27	83.13	78.24
6. BEST MATCH	50.97	46.66	48.72	86.19	82.70	84.41	**78.44**
7. WEAK MATCH	47.46	66.72	**55.47**	70.36	88.05	78.22	71.21

Table 3: CISTELL results varying the corpus language.

also many spurious links that are not duly penalized. The resulting output is not very desirable.[12] In contrast, B^3 is more P-oriented and scores conservative outputs like HEAD MATCH and BEST MATCH first, even if R is low. CEAF achieves a better compromise between P and R, as corroborated by the quality of the output.

The baselines and the system runs perform very similarly in the two corpora, but slightly better for English. It seems that language-specific issues do not result in significant differences—at least for English and Spanish—once the feature set has been appropriately adapted, e.g., including features about zero subjects or removing those about possessive phrases. Comparing the feature ranks, we find that the features that work best for each language largely overlap and are language independent, like head match, is-a match, and whether the mentions are pronominal.

5 Parameter 2: Annotation Scheme

In the second experiment, we used the 100k-word portion (from the TDT collection) shared by the OntoNotes and ACE corpora (330 OntoNotes documents occurred as 22 ACE-2003 documents, 185 ACE-2004 documents, and 123 ACE-2005 documents). CISTELL was trained on the same texts in both corpora and applied to the remainder. The three measures were then applied to each result.

Datasets Since the two annotation schemes differ significantly, we made the results comparable by mapping the ACE entities (the simpler scheme) onto the information contained in OntoNotes.[13] The mapping allowed us to focus exclusively on the differences expressed on both corpora: the types of mentions that were annotated, the definition of identity of reference, etc.

Table 4 presents the statistics for the OntoNotes dataset merged with the ACE entities. The mapping was not straightforward due to several problems: there was no match for some mentions due to syntactic or spelling reasons (e.g., *El Popo* in OntoNotes vs. *Ell Popo* in ACE). ACE mentions for which there was no parse tree node in the OntoNotes gold-standard tree were omitted, as creating a new node could have damaged the tree.

Given that only seven entity types are annotated in ACE, the number of OntoNotes mentions is al-

[12] Due to space constraints, the actual output cannot be shown here. We are happy to send it to interested requesters.

[13] Both ACE entities and types were mapped onto the OntoNotes dataset.

		#docs	#words	#mentions	#entities (e)	#singleton e	#multi-mention e
OntoNotes	Training	297	87,068	22,127	15,983	13,587	2,396
	Test	33	9,763	2,463	1,790	1,476	314
ACE	Training	297	87,068	12,951	5,873	3,599	2,274
	Test	33	9,763	1,464	746	459	287

Table 4: Corpus statistics for the aligned portion of ACE and OntoNotes on gold-standard data.

	MUC			B^3			CEAF
	P	R	F	P	R	F	P / R / F
OntoNotes scheme							
1. ALL SINGLETONS	–	–	–	100	72.68	84.18	72.68
2. HEAD MATCH	55.14	39.08	45.74	90.65	80.87	**85.48**	76.05
3. HEAD MATCH + PRON	47.10	53.05	49.90	82.28	83.13	82.70	75.15
4. STRONG MATCH	46.81	53.34	49.86	80.47	83.54	81.97	76.78
5. SUPER STRONG MATCH	46.51	40.56	43.33	84.95	80.16	82.48	76.70
6. BEST MATCH	52.47	47.40	49.80	86.10	82.80	84.42	**77.87**
7. WEAK MATCH	47.91	64.64	**55.03**	71.73	87.46	78.82	71.74
ACE scheme							
1. ALL SINGLETONS	–	–	–	100	50.96	67.51	50.96
2. HEAD MATCH	82.35	39.00	52.93	95.27	64.05	**76.60**	66.46
3. HEAD MATCH + PRON	70.11	53.90	60.94	86.49	68.20	76.27	68.44
4. STRONG MATCH	64.21	64.21	64.21	76.92	73.54	75.19	**70.01**
5. SUPER STRONG MATCH	60.51	56.55	58.46	76.71	69.19	72.76	66.87
6. BEST MATCH	67.50	56.69	61.62	82.18	71.67	76.57	69.88
7. WEAK MATCH	63.52	80.50	**71.01**	59.76	86.36	70.64	64.21

Table 5: CISTELL results varying the annotation scheme on gold-standard data.

most twice as large as the number of ACE mentions. Unlike OntoNotes, ACE mentions include premodifiers (e.g., *state* in *state lines*), national adjectives (e.g., *Iraqi*) and relative pronouns (e.g., *who, that*). Also, given that ACE entities correspond to types that are usually coreferred (e.g., people, organizations, etc.), singletons only represent 61% of all entities, while they are 85% in OntoNotes. The average entity size is 4 in ACE and 3.5 in OntoNotes.

A second major difference is the definition of coreference relations, illustrated here:

(2) [This] was [an all-white, all-Christian community that all the sudden was taken over ... by different groups].

(3) [[Mayor] John Hyman] has a simple answer.

(4) [Postville] now has 22 different nationalities ... For those who prefer [the old Postville], Mayor John Hyman has a simple answer.

In ACE, nominal predicates corefer with their subject (2), and appositive phrases corefer with the noun they are modifying (3). In contrast, they do not fall under the identity relation in OntoNotes, which follows the linguistic understanding of coreference according to which nominal predicates and appositives express properties of an entity rather than refer to a second (coreferent) entity (van Deemter and Kibble, 2000). Finally, the two schemes frequently disagree on borderline cases in which coreference turns out to be especially complex (4). As a result, some features will behave differently, e.g., the appositive feature has the opposite effect in the two datasets.

Results and Discussion From the differences pointed out above, the results shown in Table 5 might be surprising at first. Given that OntoNotes is not restricted to any semantic type and is based on a more sophisticated definition of coreference, one would not expect a system to perform better on it than on ACE. The explanation is given by the ALL SINGLETONS baseline, which is 73–84% for OntoNotes and only 51–68% for ACE. The fact that OntoNotes contains a much larger number of singletons—as Table 4 shows—results in an initial boost of performance (except with the MUC score, which ignores singletons). In contrast, the score improvement achieved by HEAD MATCH is much more noticeable on ACE than on OntoNotes, which indicates that many of its coreferent mentions share the same head.

The systematic biases of the measures that were observed in Table 3 appear again in the case of

MUC and B^3. CEAF is divided between BEST MATCH and STRONG MATCH. The higher value of the MUC score for ACE is another indication of its tendency to reward correct links much more than to penalize spurious ones (ACE has a larger proportion of multi-mention entities).

The feature rankings obtained for each dataset generally coincide as to which features are ranked best (namely NE match, is-a match, and head match), but differ in their particular ordering.

It is also possible to compare the OntoNotes results in Tables 3 and 5, the only difference being that the first training set was three times larger. Contrary to expectation, the model trained on a larger dataset performs just slightly better. The fact that more training data does not necessarily lead to an increase in performance conforms to the observation that there appear to be few general rules (e.g., head match) that systematically govern coreference relationships; rather, coreference appeals to individual unique phenomena appearing in each context, and thus after a point adding more training data does not add much new generalizable information. Pragmatic information (discourse structure, world knowledge, etc.) is probably the key, if ever there is a way to encode it.

6 Parameter 3: Preprocessing

The goal of the third experiment was to determine how much the source and nature of preprocessing information matters. Since it is often stated that coreference resolution depends on many levels of analysis, we again compared the two corpora, which differ in the amount and correctness of such information. However, in this experiment, entity mapping was applied in the opposite direction: the OntoNotes entities were mapped onto the automatically preprocessed ACE dataset. This exposes the shortcomings of automated preprocessing in ACE for identifying all the mentions identified and linked in OntoNotes.

Datasets The ACE data was morphologically annotated with a tokenizer based on manual rules adapted from the one used in CoNLL (Tjong Kim Sang and De Meulder, 2003), with TnT 2.2, a trigram POS tagger based on Markov models (Brants, 2000), and with the built-in WordNet lemmatizer (Fellbaum, 1998). Syntactic chunks were obtained from YamCha 1.33, an SVM-based NP-chunker (Kudoh and Matsumoto, 2000), and parse trees from Malt Parser 0.4, an SVM-based parser (Hall et al., 2007).

Although the number of words in Tables 4 and 6 should in principle be the same, the latter contains fewer words as it lacks the null elements (traces, ellipsed material, etc.) manually annotated in OntoNotes. Missing parse tree nodes in the automatically parsed data account for the considerably lower number of OntoNotes mentions (approx. 5,700 fewer mentions).[14] However, the proportions of singleton:multi-mention entities as well as the average entity size do not vary.

Results and Discussion The ACE scores for the automatically preprocessed models in Table 7 are about 3% lower than those based on OntoNotes gold-standard data in Table 5, providing evidence for the advantage offered by gold-standard preprocessing information. In contrast, the similar—if not higher—scores of OntoNotes can be attributed to the use of the annotated ACE entity types. The fact that these are annotated not only for proper nouns (as predicted by an automatic NER) but also for pronouns and full NPs is a very helpful feature for a coreference resolution system.

Again, the scoring metrics exhibit similar biases, but note that CEAF prefers HEAD MATCH + PRON in the case of ACE, which is indicative of the noise brought by automatic preprocessing.

A further insight is offered from comparing the feature rankings with gold-standard syntax to that with automatic preprocessing. Since we are evaluating now on the ACE data, the NE match feature is also ranked first for OntoNotes. Head and is-a match are still ranked among the best, yet syntactic features are not. Instead, features like NP type have moved further up. This reranking probably indicates that if there is noise in the syntactic information due to automatic tools, then morphological and syntactic features switch their positions.

Given that the noise brought by automatic preprocessing can be harmful, we tried leaving out the grammatical function feature. Indeed, the results increased about 2–3%, STRONG MATCH scoring the highest. This points out that conclusions drawn from automatically preprocessed data about the kind of knowledge relevant for coreference resolution might be mistaken. Using the most successful basic features can lead to the best results when only automatic preprocessing is available.

[14] In order to make the set of mentions as similar as possible to the set in Section 5, OntoNotes singletons were mapped from the ones detected in the gold-standard treebank.

		#docs	#words	#mentions	#entities (e)	#singleton e	#multi-mention e
OntoNotes	Training	297	80,843	16,945	12,127	10,253	1,874
	Test	33	9,073	1,931	1,403	1,156	247
ACE	Training	297	80,843	13,648	6,041	3,652	2,389
	Test	33	9,073	1,537	775	475	300

Table 6: Corpus statistics for the aligned portion of ACE and OntoNotes on automatically parsed data.

	MUC			B^3			CEAF
	P	R	F	P	R	F	P / R / F
OntoNotes scheme							
1. ALL SINGLETONS	–	–	–	100	72.66	84.16	72.66
2. HEAD MATCH	56.76	35.80	43.90	92.18	80.52	**85.95**	76.33
3. HEAD MATCH + PRON	47.44	54.36	50.66	82.08	83.61	82.84	74.83
4. STRONG MATCH	52.66	58.14	55.27	83.11	85.05	84.07	**78.30**
5. SUPER STRONG MATCH	51.67	46.78	49.11	85.74	82.07	83.86	77.67
6. BEST MATCH	54.38	51.70	53.01	86.00	83.60	84.78	78.15
7. WEAK MATCH	49.78	64.58	**56.22**	75.63	87.79	81.26	74.62
ACE scheme							
1. ALL SINGLETONS	–	–	–	100	50.42	67.04	50.42
2. HEAD MATCH	81.25	39.24	52.92	94.73	63.82	**76.26**	65.97
3. HEAD MATCH + PRON	69.76	53.28	60.42	86.39	67.73	75.93	**68.05**
4. STRONG MATCH	58.85	58.92	58.89	73.36	70.35	71.82	66.30
5. SUPER STRONG MATCH	56.19	50.66	53.28	75.54	66.47	70.72	63.96
6. BEST MATCH	63.38	49.74	55.74	80.97	68.11	73.99	65.97
7. WEAK MATCH	60.22	78.48	**68.15**	55.17	84.86	66.87	59.08

Table 7: CISTELL results varying the annotation scheme on automatically preprocessed data.

7 Conclusion

Regarding evaluation, the results clearly expose the systematic tendencies of the evaluation measures. The way each measure is computed makes it biased towards a specific model: MUC is generally too lenient with spurious links, B^3 scores too high in the presence of a large number of singletons, and CEAF does not agree with either of them. It is a cause for concern that they provide contradictory indications about the core of coreference, namely the resolution models—for example, the model ranked highest by B^3 in Table 7 is ranked *lowest* by MUC. We always assume evaluation measures provide a 'true' reflection of our approximation to a gold standard in order to guide research in system development and tuning.

Further support to our claims comes from the results of SemEval-2010 Task 1 (Recasens et al., 2010). The performance of the six participating systems shows similar problems with the evaluation metrics, and the singleton baseline was hard to beat even by the highest-performing systems.

Since the measures imply different conclusions about the nature of the corpora and the preprocessing information applied, should we use them now to constrain the ways our corpora are created in the first place, and what preprocessing we include or omit? Doing so would seem like circular reasoning: it invalidates the notion of the existence of a true and independent gold standard. But if apparently incidental aspects of the corpora can have such effects—effects rated quite differently by the various measures—then we have no fixed ground to stand on.

The worrisome fact that there is currently no clearly preferred and 'correct' evaluation measure for coreference resolution means that we cannot draw definite conclusions about coreference resolution systems at this time, unless they are compared on exactly the same corpus, preprocessed under the same conditions, and all three measures agree in their rankings.

Acknowledgments

We thank Dr. M. Antònia Martí for her generosity in allowing the first author to visit ISI to work with the second. Special thanks to Edgar Gonzàlez for his kind help with conversion issues.

This work was partially supported by the Spanish Ministry of Education through an FPU scholarship (AP2006-00994) and the TEXT-MESS 2.0 Project (TIN2009-13391-C04-04).

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC 1998 Workshop on Linguistic Coreference*, pages 563–566, Granada, Spain.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of EMNLP 2008*, pages 294–303, Honolulu, Hawaii.

Thorsten Brants. 2000. TnT – A statistical part-of-speech tagger. In *Proceedings of ANLP 2000*, Seattle, WA.

Aron Culotta, Michael Wick, Robert Hall, and Andrew McCallum. 2007. First-order probabilistic models for coreference resolution. In *Proceedings of HLT-NAACL 2007*, pages 81–88, Rochester, New York.

Walter Daelemans and Antal Van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press.

Pascal Denis and Jason Baldridge. 2009. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42:87–96.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The Automatic Content Extraction (ACE) Program - Tasks, Data, and Evaluation. In *Proceedings of LREC 2004*, pages 837–840.

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

Jenny Rose Finkel and Christopher D. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proceedings of ACL-HLT 2008*, pages 45–48, Columbus, Ohio.

Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of EMNLP 2009*, pages 1152–1161, Singapore. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL 2007*, pages 933–939.

Lynette Hirschman and Nancy Chinchor. 1997. MUC-7 Coreference Task Definition – Version 3.0. In *Proceedings of MUC-7*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of HLT-NAACL 2006*, pages 57–60.

Taku Kudoh and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL 2000 and LLL 2000*, pages 142–144, Lisbon, Portugal.

Xiaoqiang Luo and Imed Zitouni. 2005. Multi-lingual coreference resolution with syntactic features. In *Proceedings of HLT-EMNLP 2005*, pages 660–667, Vancouver.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proceedings of ACL 2004*, pages 21–26, Barcelona.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of HLT-EMNLP 2005*, pages 25–32, Vancouver.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL 2002*, pages 104–111, Philadelphia.

Vincent Ng. 2009. Graph-cut-based anaphoricity determination for coreference resolution. In *Proceedings of NAACL-HLT 2009*, pages 575–583, Boulder, Colorado.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of EMNLP 2008*, pages 650–659, Honolulu, Hawaii.

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. In *Proceedings of ICSC 2007*, pages 517–526, Washington, DC.

Marta Recasens and Eduard Hovy. 2009. A Deeper Look into Features for Coreference Resolution. In S. Lalitha Devi, A. Branco, and R. Mitkov, editors, *Anaphora Processing and Applications (DAARC 2009)*, volume 5847 of *LNAI*, pages 29–42. Springer-Verlag.

Marta Recasens and M. Antònia Martí. 2009. AnCora-CO: Coreferentially annotated corpora for Spanish and Catalan. *Language Resources and Evaluation*, DOI 10.1007/s10579-009-9108-x.

Marta Recasens, Lluís Màrquez, Emili Sapena, M. Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. SemEval-2010 Task 1: Coreference resolution in multiple languages. In *Proceedings of the Fifth International Workshop on Semantic Evaluations (SemEval 2010)*, Uppsala, Sweden.

Wee M. Soon, Hwee T. Ng, and Daniel C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of ACL-IJCNLP 2009*, pages 656–664, Singapore.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL 2003*, pages 142–147. Edmonton, Canada.

Olga Uryupina. 2006. Coreference resolution with and without linguistic knowledge. In *Proceedings of LREC 2006*.

Kees van Deemter and Rodger Kibble. 2000. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of MUC-6*, pages 45–52, San Francisco.

Constituency to Dependency Translation with Forests

Haitao Mi and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
P.O. Box 2704, Beijing 100190, China
{htmi,liuqun}@ict.ac.cn

Abstract

Tree-to-string systems (and their forest-based extensions) have gained steady popularity thanks to their simplicity and efficiency, but there is a major limitation: they are unable to guarantee the grammaticality of the output, which is explicitly modeled in string-to-tree systems via target-side syntax. We thus propose to combine the advantages of both, and present a novel constituency-to-dependency translation model, which uses constituency forests on the source side to direct the translation, and dependency trees on the target side (as a language model) to ensure grammaticality. Medium-scale experiments show an absolute and statistically significant improvement of +0.7 BLEU points over a state-of-the-art forest-based tree-to-string system even with fewer rules. This is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

1 Introduction

Linguistically syntax-based statistical machine translation models have made promising progress in recent years. By incorporating the syntactic annotations of parse trees from *both* or *either* side(s) of the bitext, they are believed better than phrase-based counterparts in reorderings. Depending on the type of input, these models can be broadly divided into two categories (see Table 1): the *string-based* systems whose input is a string to be simultaneously parsed and translated by a synchronous grammar, and the *tree-based* systems whose input is already a parse tree to be directly converted into a target tree or string. When we also take into account the type of output (tree or string), the *tree-based* systems can be divided into *tree-to-string* and *tree-to-tree* efforts.

tree on	*examples (partial)*	*fast*	*gram.*	BLEU
source	Liu06, Huang06	+	-	+
target	Galley06, Shen08	-	+	+
both	Ding05, Liu09	+	+	-
both	**our work**	+	+	+

Table 1: A classification and comparison of linguistically syntax-based SMT systems, where *gram.* denotes grammaticality of the output.

On one hand, tree-to-string systems (Liu et al., 2006; Huang et al., 2006) have gained significant popularity, especially after incorporating packed forests (Mi et al., 2008; Mi and Huang, 2008; Liu et al., 2009; Zhang et al., 2009). Compared with their string-based counterparts, tree-based systems are much faster in decoding (linear time vs. cubic time, see (Huang et al., 2006)), do not require a binary-branching grammar as in string-based models (Zhang et al., 2006; Huang et al., 2009), and can have separate grammars for parsing and translation (Huang et al., 2006). However, they have a major limitation that they do not have a principled mechanism to guarantee grammaticality on the target side, since there is no linguistic tree structure of the output.

On the other hand, string-to-tree systems explicitly model the grammaticality of the output by using target syntactic trees. Both string-to-constituency system (e.g., (Galley et al., 2006; Marcu et al., 2006)) and string-to-dependency model (Shen et al., 2008) have achieved significant improvements over the state-of-the-art formally syntax-based system Hiero (Chiang, 2007). However, those systems also have some limitations that they run slowly (in cubic time) (Huang et al., 2006), and do not utilize the useful syntactic information on the source side.

We thus combine the advantages of both tree-to-string and string-to-tree approaches, and propose

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1433–1442,
Uppsala, Sweden, 11-16 July 2010.

a novel constituency-to-dependency model, which uses constituency forests on the source side to direct translation, and dependency trees on the target side to guarantee grammaticality of the output. In contrast to conventional tree-to-tree approaches (Ding and Palmer, 2005; Quirk et al., 2005; Xiong et al., 2007; Zhang et al., 2007; Liu et al., 2009), which only make use of a single type of trees, our model is able to combine two types of trees, outperforming both phrase-based and tree-to-string systems. Current tree-to-tree models (Xiong et al., 2007; Zhang et al., 2007; Liu et al., 2009) still have not outperformed the phrase-based system Moses (Koehn et al., 2007) significantly even with the help of forests.[1]

Our new constituency-to-dependency model (Section 2) extracts rules from word-aligned pairs of source constituency forests and target dependency trees (Section 3), and translates source constituency forests into target dependency trees with a set of features (Section 4). Medium data experiments (Section 5) show a statistically significant improvement of +0.7 BLEU points over a state-of-the-art forest-based tree-to-string system even with less translation rules, this is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

2 Model

Figure 1 shows a word-aligned source constituency forest F_c and target dependency tree D_e, our constituency to dependency translation model can be formalized as:

$$\begin{aligned} \mathrm{P}(F_c, D_e) &= \sum_{C_c \in F_c} \mathrm{P}(C_c, D_e) \\ &= \sum_{C_c \in F_c} \sum_{o \in O} \mathrm{P}(O) \\ &= \sum_{C_c \in F_c} \sum_{o \in O} \prod_{r \in o} \mathrm{P}(r), \end{aligned} \quad (1)$$

where C_c is a constituency tree in F_c, o is a derivation that translates C_c to D_e, O is the set of derivation, r is a constituency to dependency translation rule.

[1] According to the reports of Liu et al. (2009), their forest-based constituency-to-constituency system achieves a comparable performance against Moses (Koehn et al., 2007), but a significant improvement of +3.6 BLEU points over the 1-best tree-based constituency-to-constituency system.

2.1 Constituency Forests on the Source Side

A constituency forest (in Figure 1 left) is a compact representation of all the derivations (i.e., parse trees) for a given sentence under a context-free grammar (Billot and Lang, 1989).

More formally, following Huang (2008), such a constituency forest is a pair $F_c = G^f = \langle V^f, H^f \rangle$, where V^f is the set of **nodes**, and H^f the set of **hyperedges**. For a given source sentence $c_{1:m} = c_1 \dots c_m$, each node $v^f \in V^f$ is in the form of $X_{i,j}$, which denotes the recognition of nonterminal X spanning the substring from positions i through j (that is, $c_{i+1} \dots c_j$). Each hyperedge $h^f \in H^f$ is a pair $\langle tails(h^f), head(h^f) \rangle$, where $head(h^f) \in V^f$ is the **consequent node** in the deductive step, and $tails(h^f) \in (V^f)^*$ is the list of **antecedent nodes**. For example, the hyperedge h_0^f in Figure 1 for deduction (*)

$$\frac{\mathrm{NPB}_{0,1} \quad \mathrm{CC}_{1,2} \quad \mathrm{NPB}_{2,3}}{\mathrm{NP}_{0,3}}, \quad (*)$$

is notated:

$$\langle (\mathrm{NPB}_{0,1},\ \mathrm{CC}_{1,2},\ \mathrm{NPB}_{2,3}),\ \ \mathrm{NP}_{0,3} \rangle.$$

where

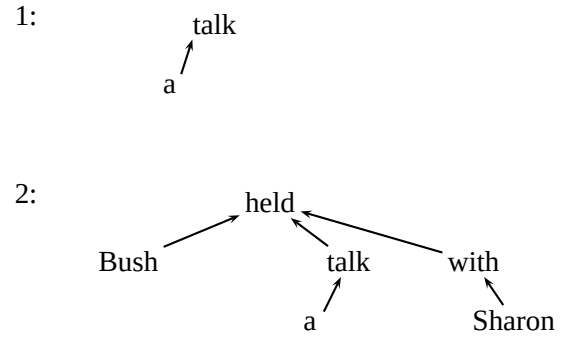
$$head(h_0^f) = \{\mathrm{NP}_{0,3}\},$$
and
$$tails(h_0^f) = \{\mathrm{NPB}_{0,1}, \mathrm{CC}_{1,2}, \mathrm{NPB}_{2,3}\}.$$

The solid line in Figure 1 shows the best parse tree, while the dashed one shows the second best tree. Note that common sub-derivations like those for the verb $\mathrm{VPB}_{3,5}$ are shared, which allows the forest to represent exponentially many parses in a compact structure.

We also denote $IN(v^f)$ to be the set of **incoming hyperedges** of node v^f, which represents the different ways of deriving v^f. Take node $\mathrm{IP}_{0,5}$ in Figure 1 for example, $IN(\mathrm{IP}_{0,5}) = \{h_1^f, h_2^f\}$. There is also a distinguished **root node** TOP in each forest, denoting the goal item in parsing, which is simply $\mathrm{S}_{0,m}$ where S is the start symbol and m is the sentence length.

2.2 Dependency Trees on the Target Side

A dependency tree for a sentence represents each word and its syntactic dependents through directed arcs, as shown in the following examples. The main advantage of a dependency tree is that it can explore the long distance dependency.



We use the lexicon dependency grammar (Hellwig, 2006) to express a projective dependency tree. Take the dependency trees above for example, they will be expressed:

1: (a) talk

2: (Bush) held ((a) talk) (with (Sharon))

where the lexicons in brackets represent the dependencies, while the lexicon out the brackets is the head.

More formally, a dependency tree is also a pair $D_e = G^d = \langle V^d, H^d \rangle$. For a given target sentence $e_{1:n} = e_1 \dots e_n$, each node $v^d \in V^d$ is a word e_i ($1 \leqslant i \leqslant n$), each hyperedge $h^d \in H^d$ is a directed arc $\langle v_i^d, v_j^d \rangle$ from node v_i^d to its head node v_j^d. Following the formalization of the constituency forest scenario, we denote a pair $\langle tails(h^d), head(h^d) \rangle$ to be a hyperedge h^d, where $head(h^d)$ is the head node, $tails(h^d)$ is the node where h^d leaves from.

We also denote $L_l(v^d)$ and $L_r(v^d)$ to be the left and right children sequence of node v^d from the nearest to the farthest respectively. Take the node $v_2^d =$ "held" for example:

$$L_l(v_2^d) = \{\text{Bush}\},$$
$$L_r(v_2^d) = \{\text{talk, with}\}.$$

2.3 Hypergraph

Actually, both the constituency forest and the dependency tree can be formalized as a **hypergraph** G, a pair $\langle V, H \rangle$. We use G^f and G^d to distinguish them. For simplicity, we also use F_c and D_e to denote a constituency forest and a dependency tree respectively. Specifically, the size of $tails(h^d)$ of a hyperedge h^d in a dependency tree is a constant one.

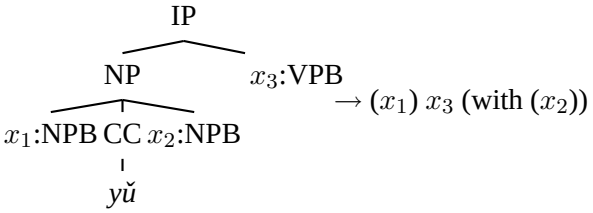


Figure 2: Example of the rule r_1. The Chinese conjunction *yǔ* "and" is translated into English preposition "with".

3 Rule Extraction

We extract constituency to dependency rules from word-aligned source constituency forest and target dependency tree pairs (Figure 1). We mainly extend the tree-to-string rule extraction algorithm of Mi and Huang (2008) to our scenario. In this section, we first formalize the constituency to string translation rule (Section 3.1). Then we present the restrictions for dependency structures as well formed fragments (Section 3.2). Finally, we describe our rule extraction algorithm (Section 3.3), fractional counts computation and probabilities estimation (Section 3.4).

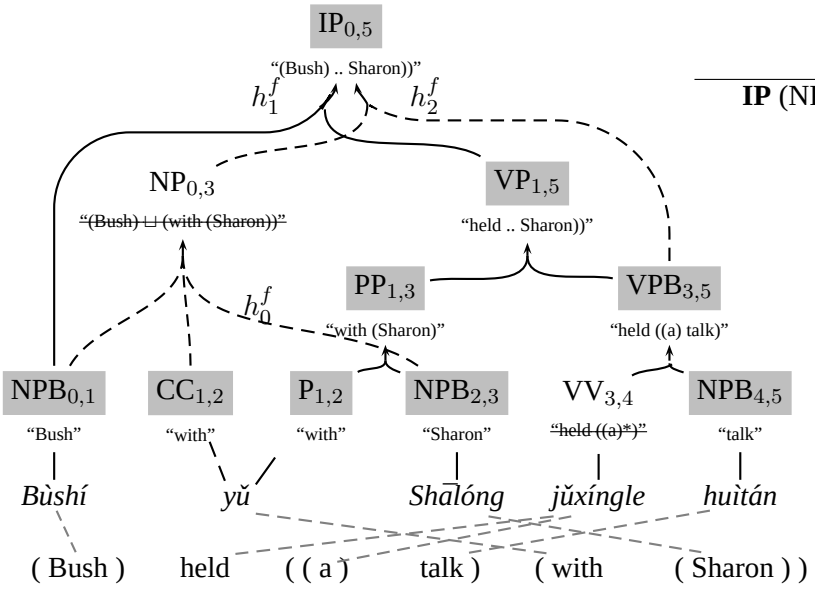
3.1 Constituency to Dependency Rule

More formally, a **constituency to dependency translation rule** r is a tuple $\langle lhs(r), rhs(r), \phi(r) \rangle$, where $lhs(r)$ is the source side tree fragment, whose internal nodes are labeled by nonterminal symbols (like NP and VP), and whose frontier nodes are labeled by source language words c_i (like "*yǔ*") or variables from a set $\mathcal{X} = \{x_1, x_2, \dots\}$; $rhs(r)$ is expressed in the target language dependency structure with words e_j (like "with") and variables from the set $\mathcal{X}$; and $\phi(r)$ is a mapping from $\mathcal{X}$ to nonterminals. Each variable $x_i \in \mathcal{X}$ occurs *exactly once* in $lhs(r)$ and *exactly once* in $rhs(r)$. For example, the rule r_1 in Figure 2,

$lhs(r_1) =$ IP(NP(x_1 CC(*yǔ*) x_2) x_3),
$rhs(r_1) = (x_1)\ x_3\ (\text{with}\ (x_2))$,
$\phi(r_1) = \{x_1 \mapsto \text{NPB}, x_2 \mapsto \text{NPB}, x_3 \mapsto \text{VPB}\}$.

3.2 Well Formed Dependency Fragment

Following Shen et al. (2008), we also restrict $rhs(r)$ to be **well formed** dependency fragment. The main difference between us is that we use more flexible restrictions. Given a dependency



Minimal rules extracted

IP (NP(x_1:NPB x_2:CC x_3:NPB) x_4:VPB) → (x_1) x_4 (x_2 (x_3))

IP (x_1:NPB x_2:VP) → (x_1) x_2

VP (x_1:PP x_2:VPB) → x_2 (x_1)

PP (x_1:P x_2:NPB) → x_1 (x_2)

VPB (VV(*jǔxíngle*)) x_1:NPB) → held ((a) x_1)

NPB (*Bùshí*) → Bush

NPB (*huìtán*) → talk

CC (*yǔ*) → with

P (*yǔ*) → with

NPB (*Shālóng*) → Sharon

Figure 1: Forest-based constituency to dependency rule extraction.

fragment $d_{i:j}$ composed by the words from i to j, two kinds of well formed structures are defined as follows:

Fixed on one node v^d_{one}, **fixed** for short, if it meets the following conditions:

- the head of v^d_{one} is out of $[i, j]$, i.e.: $\forall h^d$, if $tails(h^d) = v^d_{one} \Rightarrow head(h^d) \notin e_{i:j}$.
- the heads of other nodes except v^d_{one} are in $[i, j]$, i.e.: $\forall k \in [i, j]$ and $v^d_k \neq v^d_{one}, \forall h^d$ if $tails(h^d) = v^d_k \Rightarrow head(h^d) \in e_{i:j}$.

Floating with multi nodes M, **floating** for short, if it meets the following conditions:

- all nodes in M have a same head node, i.e.: $\exists x \notin [i, j], \forall h^d$ if $tails(h^d) \in M \Rightarrow head(h^d) = v^h_x$.
- the heads of other nodes not in M are in $[i, j]$, i.e.: $\forall k \in [i, j]$ and $v^d_k \notin M, \forall h^d$ if $tails(h^d) = v^d_k \Rightarrow head(h^d) \in e_{i:j}$.

Take the " (Bush) held ((a) talk))(with (Sharon)) " for example: partial fixed examples are " (Bush) held " and " held ((a) talk)"; while the partial floating examples are " (talk) (with (Sharon)) " and " ((a) talk) (with (Sharon)) ". Please note that the floating structure " (talk) (with (Sharon)) " can not be allowed in Shen et al. (2008)'s model.

The dependency structure " held ((a))" is not a well formed structure, since the head of word "a" is out of scope of this structure.

3.3 Rule Extraction Algorithm

The algorithm shown in this Section is mainly extended from the forest-based tree-to-string extraction algorithm (Mi and Huang, 2008). We extract rules from word-aligned source constituency forest and target dependency tree pairs (see Figure 1) in three steps:

(1) frontier set computation,

(2) fragmentation,

(3) composition.

The **frontier set** (Galley et al., 2004) is the potential points to "cut" the forest and dependency tree pair into fragments, each of which will form a **minimal rule** (Galley et al., 2006).

However, not every fragment can be used for rule extraction, since it may or may not respect to the restrictions, such as word alignments and well formed dependency structures. So we say a fragment is **extractable** if it respects to all restrictions. The root node of every extractable tree fragment corresponds to a **faithful structure** on the target side, in which case there is a "translational equivalence" between the subtree rooted at the node and the corresponding target structure. For example, in Figure 1, every node in the forest is annotated with its corresponding English structure. The $NP_{0,3}$ node maps to a non-contiguous structure "(Bush) ⊔ (with (Sharon))", the $VV_{3,4}$ node maps to a contiguous but non-faithful structure "held ((a) *)".

Algorithm 1 Forest-based constituency to dependency rule extraction.

Input: Source constituency forest F_c, target dependency tree D_e, and alignment a
Output: Minimal rule set $\mathcal{R}$

1: $fs \leftarrow$ FRONTIER(F_c, D_e, a) ▷ compute frontier set
2: **for** each $v^f \in fs$ **do**
3: $open \leftarrow \{\langle \varnothing, \{v^f\}\rangle\}$ ▷ initial queue of growing fragments
4: **while** $open \neq \varnothing$ **do**
5: $\langle hs, exps\rangle \leftarrow open.pop()$ ▷ extract a fragment
6: **if** $exps = \varnothing$ **then** ▷ nothing to expand?
7: generate a rule r using fragment hs ▷ generate a rule
8: $\mathcal{R}$.append(r)
9: **else** ▷ incomplete: further expand
10: $v' \leftarrow exps$.pop() ▷ a non-frontier node
11: **for** each $h^f \in IN(v')$ **do**
12: $newexps \leftarrow exps \cup (tails(h^f) \setminus fs)$ ▷ expand
13: $open$.append($\langle hs \cup \{h^f\}, newexps\rangle$)

Following Mi and Huang (2008), given a source target sentence pair $\langle c_{1:m}, e_{1:n}\rangle$ with an alignment a, the **span** of node v^f on source forest is the set of target words aligned to leaf nodes under v^f:

$$span(v^f) \triangleq \{e_i \in e_{1:n} \mid \exists c_j \in yield(v^f), (c_j, e_i) \in a\}.$$

where the $yield(v^f)$ is all the leaf nodes under v^f. For each $span(v^f)$, we also denote $dep(v^f)$ to be its corresponding dependency structure, which represents the dependency structure of all the words in $span(v^f)$. Take the $span(\text{PP}_{1,3})$ ={with, Sharon} for example, the corresponding $dep(\text{PP}_{1,3})$ is "with (Sharon)". A $dep(v^f)$ is **faithful structure** to node v^f if it meets the following restrictions:

- all words in $span(v^f)$ form a continuous substring $e_{i:j}$,
- every word in $span(v^f)$ is *only* aligned to leaf nodes of v^f, i.e.: $\forall e_i \in span(v^f), (c_j, e_i) \in a \Rightarrow c_j \in yield(v^f)$,
- $dep(v^f)$ is a well formed dependency structure.

For example, node $\text{VV}_{3,4}$ has a non-faithful structure (crossed out in Figure 1), since its $dep(\text{VV}_{3,4} =$ " held ((a) *)" is not a well formed structure, where the head of word "a" lies in the outside of its words covered. Nodes with faithful structure form the **frontier set** (shaded nodes in Figure 1) which serve as potential cut points for rule extraction.

Given the frontier set, fragmentation step is to "cut" the forest at all frontier nodes and form tree fragments, each of which forms a rule with variables matching the frontier descendant nodes. For example, the forest in Figure 1 is cut into 10 pieces, each of which corresponds to a minimal rule listed on the right.

Our rule extraction algorithm is formalized in Algorithm 1. After we compute the frontier set *fs* (line 1). We visit each frontier node $v^f \in fs$ on the source constituency forest F_c, and keep a queue $open$ of growing fragments rooted at v^f. We keep expanding incomplete fragments from $open$, and extract a rule if a complete fragment is found (line 7). Each fragment *hs* in $open$ is associated with a list of *expansion sites* ($exps$ in line 5) being the subset of leaf nodes of the current fragment that are *not* in the frontier set. So each fragment along hyperedge h is associated with

$$exps = tails(h^f) \setminus fs.$$

A fragment is complete if its expansion sites is empty (line 6), otherwise we pop one expansion node v' to grow and spin-off new fragments by following hyperedges of v', adding new expansion sites (lines 11-13), until all active fragments are complete and $open$ queue is empty (line 4).

After we get all the minimal rules, we glue them together to form **composed rule**s following Galley et al. (2006). For example, the composed rule r_1 in Figure 2 is glued by the following two minimal rules:

IP (NP(x_1:NPB x_2:CC x_3:NPB) x_4:VPB) $\rightarrow$ (x_1) x_4 (x_2 (x_3)) r_2

CC (*yǔ*) $\rightarrow$ with r_3

where x_2:CC in r_2 is replaced with r_3 accordingly.

3.4 Fractional Counts and Rule Probabilities

Following Mi and Huang (2008), we penalize a rule r by the posterior probability of the corresponding constituent tree fragment $lhs(r)$, which can be computed in an Inside-Outside fashion, being the product of the outside probability of its root node, the inside probabilities of its leaf nodes, and the probabilities of hyperedges involved in the fragment.

$$\begin{aligned}\alpha\beta(lhs(r)) =& \alpha(root(r)) \\ & \cdot \prod_{h^f \in lhs(r)} \mathrm{P}(h^f) \\ & \cdot \prod_{v^f \in leaves(lhs(r))} \beta(v^f)\end{aligned} \quad (2)$$

where $root$(r) is the root of the rule r, $\alpha(v)$ and $\beta(v)$ are the outside and inside probabilities of node v, and $leaves(lhs(r))$ returns the leaf nodes of a tree fragment lhs(r).

We use fractional counts to compute three conditional probabilities for each rule, which will be used in the next section:

$$\mathrm{P}(r \mid lhs(r)) = \frac{c(r)}{\sum_{r':lhs(r')=lhs(r)} c(r')}, \quad (3)$$

$$\mathrm{P}(r \mid rhs(r)) = \frac{c(r)}{\sum_{r':rhs(r')=rhs(r)} c(r')}, \quad (4)$$

$$\mathrm{P}(r \mid root(r)) = \frac{c(r)}{\sum_{r':root(r')=root(r)} c(r')}. \quad (5)$$

4 Decoding

Given a source forest F_c, the decoder searches for the best derivation o^* among the set of all possible derivations O, each of which forms a source side constituent tree $T_c(o)$, a target side string $e(o)$, and a target side dependency tree $D_e(o)$:

$$\begin{aligned}o^* = \arg\max_{T_c \in F_c, o \in O} & \lambda_1 \log \mathrm{P}(o \mid T_c) \\ & + \lambda_2 \log \mathrm{P}_{\mathrm{lm}}(e(o)) \\ & + \lambda_3 \log \mathrm{P}_{\mathrm{DLM_w}}(D_e(o)) \\ & + \lambda_4 \log \mathrm{P}_{\mathrm{DLM_p}}(D_e(o)) \\ & + \lambda_5 \log \mathrm{P}(T_c(o)) \\ & + \lambda_6 ill(o) + \lambda_7 |o| + \lambda_8 |e(o)|,\end{aligned} \quad (6)$$

where the first two terms are translation and language model probabilities, $e(o)$ is the target string (English sentence) for derivation o, the third and forth items are the dependency language model probabilities on the target side computed with words and POS tags separately, $D_e(o)$ is the target dependency tree of o, the fifth one is the parsing probability of the source side tree $T_c(o) \in F_c$, the $ill(o)$ is the penalty for the number of ill-formed dependency structures in o, and the last two terms are derivation and translation length penalties, respectively. The conditional probability $\mathrm{P}(o \mid T_c)$ is decomposes into the product of rule probabilities:

$$\mathrm{P}(o \mid T_c) = \prod_{r \in o} \mathrm{P}(r), \quad (7)$$

where each $\mathrm{P}(r)$ is the product of five probabilities:

$$\begin{aligned}\mathrm{P}(r) =& \mathrm{P}(r \mid lhs(r))^{\lambda_9} \cdot \mathrm{P}(r \mid rhs(r))^{\lambda_{10}} \\ & \cdot \mathrm{P}(r \mid root(lhs(r)))^{\lambda_{11}} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(lhs(r) \mid rhs(r))^{\lambda_{12}} \\ & \cdot \mathrm{P}_{\mathrm{lex}}(rhs(r) \mid lhs(r))^{\lambda_{13}},\end{aligned} \quad (8)$$

where the first three are conditional probabilities based on fractional counts of rules defined in Section 3.4, and the last two are lexical probabilities. When computing the lexical translation probabilities described in (Koehn et al., 2003), we only take into accout the terminals in a rule. If there is no terminal, we set the lexical probability to 1.

The decoding algorithm works in a bottom-up search fashion by traversing each node in forest F_c. We first use pattern-matching algorithm of Mi et al. (2008) to convert F_c into a **translation forest**, each hyperedge of which is associated with a constituency to dependency translation rule. However, pattern-matching failure[2] at a node v^f will

[2]Pattern-matching failure at a node v^f means there is no translation rule can be matched at v^f or no translation hyperedge can be constructed at v^f.

cut the derivation path and lead to translation failure. To tackle this problem, we construct a **pseudo translation rule** for each parse hyperedge $h^f \in IN(v^f)$ by mapping the CFG rule into a target dependency tree using the head rules of Magerman (1995). Take the hyperedge h_0^f in Figure1 for example, the corresponding pseudo translation rule is:

NP(x_1:NPB x_2:CC x_3:NPB) $\rightarrow$ (x_1) (x_2) x_3,

since the x_3:NPB is the head word of the CFG rule: NP $\rightarrow$ NPB CC NPB.

After the translation forest is constructed, we traverse each node in translation forest also in bottom-up fashion. For each node, we use the **cube pruning** technique (Chiang, 2007; Huang and Chiang, 2007) to produce partial hypotheses and compute all the feature scores including the dependency language model score (Section 4.1). If all the nodes are visited, we trace back along the 1-best derivation at goal item $S_{0,m}$ and build a target side dependency tree. For k-best search after getting 1-best derivation, we use the lazy Algorithm 3 of Huang and Chiang (2005) that works backwards from the root node, incrementally computing the second, third, through the kth best alternatives.

4.1 Dependency Language Model Computing

We compute the score of a dependency language model for a dependency tree D_e in the same way proposed by Shen et al. (2008). For each nonterminal node $v_h^d = e_h$ in D_e and its children sequences $L_l = e_{l_1}, e_{l_2}...e_{l_i}$ and $L_r = e_{r_1}, e_{r_2}...e_{r_j}$, the probability of a trigram is computed as follows:

$$\mathrm{P}(L_l, L_r \mid e_h\S) = \mathrm{P}(L_l \mid e_h\S) \cdot \mathrm{P}(L_r \mid e_h\S), \quad (9)$$

where the $\mathrm{P}(L_l \mid e_h\S)$ is decomposed to be:

$$\begin{aligned} \mathrm{P}(L_l \mid e_h\S) = & \mathrm{P}(e_{l_l} \mid e_h\S) \\ & \cdot \mathrm{P}(e_{l_2} \mid e_{l_1}, e_h\S) \\ & ... \\ & \cdot \mathrm{P}(e_{l_n} \mid e_{l_{n-1}}, e_{l_{n-2}}). \end{aligned} \quad (10)$$

We use the suffix "§" to distinguish the head word and child words in the dependency language model.

In order to alleviate the problem of data sparse, we also compute a dependency language model for POS tages over a dependency tree. We store the POS tag information on the target side for each constituency-to-dependency rule. So we will also generate a POS taged dependency tree simultaneously at the decoding time. We calculate this dependency language model by simply replacing each e_i in equation 9 with its tag $t(e_i)$.

5 Experiments

5.1 Data Preparation

Our training corpus consists of 239K sentence pairs with about 6.9M/8.9M words in Chinese/English, respectively. We first word-align them by GIZA++ (Och and Ney, 2000) with refinement option "grow-diag-and" (Koehn et al., 2003), and then parse the Chinese sentences using the parser of Xiong et al. (2005) into parse forests, which are pruned into relatively small forests with a pruning threshold 3. We also parse the English sentences using the parser of Charniak (2000) into 1-best constituency trees, which will be converted into dependency trees using Magerman (1995)'s head rules. We also store the POS tag information for each word in dependency trees, and compute two different dependency language models for words and POS tags in dependency tree separately. Finally, we apply translation rule extraction algorithm described in Section 3. We use SRI Language Modeling Toolkit (Stolcke, 2002) to train a 4-gram language model with Kneser-Ney smoothing on the first 1/3 of the Xinhua portion of Gigaword corpus. At the decoding step, we again parse the input sentences into forests and prune them with a threshold 10, which will direct the translation (Section 4).

We use the 2002 NIST MT Evaluation test set as our development set and the 2005 NIST MT Evaluation test set as our test set. We evaluate the translation quality using the BLEU-4 metric (Papineni et al., 2002), which is calculated by the script mteval-v11b.pl with its default setting which is case-insensitive matching of n-grams. We use the standard minimum error-rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on development set.

5.2 Results

Table 2 shows the results on the test set. Our baseline system is a state-of-the-art forest-based constituency-to-string model (Mi et al., 2008), or *forest c2s* for short, which translates a source forest into a target string by pattern-matching the

constituency-to-string (*c2s*) rules and the bilingual phrases (*s2s*). The baseline system extracts 31.9M *c2s* rules, 77.9M *s2s* rules respectively and achieves a BLEU score of 34.17 on the test set[3].

At first, we investigate the influence of different rule sets on the performance of baseline system. We first restrict the target side of translation rules to be well-formed structures, and we extract 13.8M constituency-to-dependency (*c2d*) rules, which is 43% of *c2s* rules. We also extract 9.0M string-to-dependency (*s2d*) rules, which is only 11.6% of *s2s* rules. Then we convert *c2d* and *s2d* rules to *c2s* and *s2s* rules separately by removing the target-dependency structures and feed them into the baseline system. As shown in the third line in the column of BLEU score, the performance drops 1.7 BLEU points over baseline system due to the poorer rule coverage. However, when we further use all *s2s* rules instead of *s2d* rules in our next experiment, it achieves a BLEU score of 34.03, which is very similar to the baseline system. Those results suggest that restrictions on *c2s* rules won't hurt the performance, but restrictions on *s2s* will hurt the translation quality badly. So we should utilize all the *s2s* rules in order to preserve a good coverage of translation rule set.

The last two lines in Table 2 show the results of our new forest-based constituency-to-dependency model (*forest c2d* for short). When we only use *c2d* and *s2d* rules, our system achieves a BLEU score of 33.25, which is lower than the baseline system in the first line. But, with the same rule set, our model still outperform the result in the second line. This suggests that using dependency language model really improves the translation quality by less than 1 BLEU point.

In order to utilize all the *s2s* rules and increase the rule coverage, we parse the target strings of the *s2s* rules into dependency fragments, and construct the *pseudo s2d* rules (*s2s-dep*). Then we use *c2d* and *s2s-dep* rules to direct the translation. With the help of the dependency language model, our new model achieves a significant improvement of +0.7 BLEU points over the *forest c2s* baseline system ($p < 0.05$, using the *sign-test* suggested by Collins et al. (2005)). For the first time, a tree-to-tree model can surpass tree-to-string counterparts significantly even with fewer rules.

[3]According to the reports of Liu et al. (2009), with a more larger training corpus (FBIS plus 30K) but *no* name entity translations (+1 BLEU points if it is used), their forest-based constituency-to-constituency model achieves a BLEU score of 30.6, which is similar to Moses (Koehn et al., 2007). So our baseline system is much better than the BLEU score (30.6+1) of the constituency-to-constituency system and Moses.

System	Rule Set		BLEU
	Type	#	
forest c2s	c2s	31.9M	34.17
	s2s	77.9M	
	c2d	13.8M	32.48(↓1.7)
	s2d	9.0M	
	c2d	13.8M	34.03(↓0.1)
	s2s	77.9M	
forest c2d	c2d	13.8M	33.25(↓0.9)
	s2d	9.0M	
	c2d	13.8M	34.88(↑0.7)
	s2s-dep	77.9M	

Table 2: Statistics of different types of rules extracted on training corpus and the BLEU scores on the test set.

6 Related Work

The concept of packed forest has been used in machine translation for several years. For example, Huang and Chiang (2007) use forest to characterize the search space of decoding with integrated language models. Mi et al. (2008) and Mi and Huang (2008) use forest to direct translation and extract rules rather than 1-best tree in order to weaken the influence of parsing errors, this is also the first time to use forest directly in machine translation. Following this direction, Liu et al. (2009) and Zhang et al. (2009) apply forest into tree-to-tree (Zhang et al., 2007) and tree-sequence-to-string models(Liu et al., 2007) respectively. Different from Liu et al. (2009), we apply forest into a new constituency tree to dependency tree translation model rather than constituency tree-to-tree model.

Shen et al. (2008) present a string-to-dependency model. They define the well-formed dependency structures to reduce the size of translation rule set, and integrate a dependency language model in decoding step to exploit long distance word relations. This model shows a significant improvement over the state-of-the-art hierarchical phrase-based system (Chiang, 2005). Compared with this work, we put fewer restrictions on the definition of well-formed dependency structures in order to extract more rules; the

other difference is that we can also extract more expressive constituency to dependency rules, since the source side of our rule can encode multi-level reordering and contain more variables being larger than two; furthermore, our rules can be pattern-matched at high level, which is more reasonable than using glue rules in Shen et al. (2008)'s scenario; finally, the most important one is that our model runs very faster.

Liu et al. (2009) propose a forest-based constituency-to-constituency model, they put more emphasize on how to utilize parse forest to increase the tree-to-tree rule coverage. By contrast, we only use 1-best dependency trees on the target side to explore long distance relations and extract translation rules. Theoretically, we can extract more rules since dependency tree has the best inter-lingual phrasal cohesion properties (Fox, 2002).

7 Conclusion and Future Work

In this paper, we presented a novel forest-based constituency-to-dependency translation model, which combines the advantages of both tree-to-string and string-to-tree systems, runs fast and guarantees grammaticality of the output. To learn the constituency-to-dependency translation rules, we first identify the frontier set for all the nodes in the constituency forest on the source side. Then we fragment them and extract minimal rules. Finally, we glue them together to be composed rules. At the decoding step, we first parse the input sentence into a constituency forest. Then we convert it into a translation forest by patter-matching the constituency to string rules. Finally, we traverse the translation forest in a bottom-up fashion and translate it into a target dependency tree by incorporating string-based and dependency-based language models. Using all constituency-to-dependency translation rules and bilingual phrases, our model achieves +0.7 points improvement in BLEU score significantly over a state-of-the-art forest-based tree-to-string system. This is also the first time that a tree-to-tree model can surpass tree-to-string counterparts.

In the future, we will do more experiments on rule coverage to compare the constituency-to-constituency model with our model. Furthermore, we will replace 1-best dependency trees on the target side with dependency forests to further increase the rule coverage.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 90920004, and 863 State Key Project No. 2006AA010108. We thank the anonymous reviewers for their insightful comments. We are also grateful to Liang Huang for his valuable suggestions.

References

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of ACL '89*, pages 143–151.

Eugene Charniak. 2000. A maximum-entropy inspired parser. In *Proceedings of NAACL*, pages 132–139.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, Michigan, June.

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

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548, June.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *In Proceedings of EMNLP-02*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING-ACL*, pages 961–968, July.

Peter Hellwig. 2006. *Parsing with Dependency Grammars*, volume II. An International Handbook of Contemporary Research.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151, June.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*.

Liang Huang, Hao Zhang, Daniel Gildea, , and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Comput. Linguist.*

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

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton, Canada, May.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*, pages 177–180, June.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616, Sydney, Australia, July.

Yang Liu, Yun Huang, Qun Liu, and Shouxun Lin. 2007. Forest-to-string statistical translation rules. In *Proceedings of ACL*, pages 704–711, June.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of ACL/IJCNLP*, August.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of ACL*, pages 276–283, June.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP*, pages 44–52, July.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*, pages 206–214, Honolulu, Hawaii, October.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL-08:HLT*, pages 192–199, Columbus, Ohio, June.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*, pages 440–447.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadephia, USA, July.

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

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, June.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.

Deyi Xiong, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. In *Proceedings of IJCNLP 2005*, pages 70–81.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A dependency treelet string correspondence model for statistical machine translation. In *Proceedings of SMT*, pages 40–47.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. of HLT-NAACL*.

Min Zhang, Hongfei Jiang, Aiti Aw, Jun Sun, Sheng Li, and Chew Lim Tan. 2007. A tree-to-tree alignment-based model for statistical machine translation. In *Proceedings of MT-Summit*.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based tree sequence to string translation model. In *Proceedings of the ACL/IJCNLP 2009*.

Learning to Translate with Source and Target Syntax

David Chiang
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA
chiang@isi.edu

Abstract

Statistical translation models that try to capture the recursive structure of language have been widely adopted over the last few years. These models make use of varying amounts of information from linguistic theory: some use none at all, some use information about the grammar of the target language, some use information about the grammar of the source language. But progress has been slower on translation models that are able to learn the relationship between the grammars of both the source and target language. We discuss the reasons why this has been a challenge, review existing attempts to meet this challenge, and show how some old and new ideas can be combined into a simple approach that uses both source and target syntax for significant improvements in translation accuracy.

1 Introduction

Statistical translation models that use synchronous context-free grammars (SCFGs) or related formalisms to try to capture the recursive structure of language have been widely adopted over the last few years. The simplest of these (Chiang, 2005) make no use of information from syntactic theories or syntactic annotations, whereas others have successfully incorporated syntactic information on the target side (Galley et al., 2004; Galley et al., 2006) or the source side (Liu et al., 2006; Huang et al., 2006). The next obvious step is toward models that make full use of syntactic information on both sides. But the natural generalization to this setting has been found to underperform phrase-based models (Liu et al., 2009; Ambati and Lavie, 2008), and researchers have begun to explore solutions (Zhang et al., 2008; Liu et al., 2009).

In this paper, we explore the reasons why tree-to-tree translation has been challenging, and how source syntax and target syntax might be used together. Drawing on previous successful attempts to relax syntactic constraints during grammar extraction in various ways (Zhang et al., 2008; Liu et al., 2009; Zollmann and Venugopal, 2006), we compare several methods for extracting a synchronous grammar from tree-to-tree data. One confounding factor in such a comparison is that some methods generate many new syntactic categories, making it more difficult to satisfy syntactic constraints at decoding time. We therefore propose to move these constraints from the formalism into the model, implemented as features in the hierarchical phrase-based model Hiero (Chiang, 2005). This augmented model is able to learn from data whether to rely on syntax or not, or to revert back to monotone phrase-based translation.

In experiments on Chinese-English and Arabic-English translation, we find that when both source and target syntax are made available to the model in an unobtrusive way, the model chooses to build structures that are more syntactically well-formed and yield significantly better translations than a nonsyntactic hierarchical phrase-based model.

2 Grammar extraction

A *synchronous tree-substitution grammar* (STSG) is a set of rules or *elementary tree pairs* (γ, α), where:

- γ is a tree whose interior labels are source-language nonterminal symbols and whose frontier labels are source-language nonterminal symbols or terminal symbols (words). The nonterminal-labeled frontier nodes are called *substitution nodes*, conventionally marked with an arrow ($\downarrow$).
- α is a tree of the same form except with

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1443–1452,
Uppsala, Sweden, 11-16 July 2010.

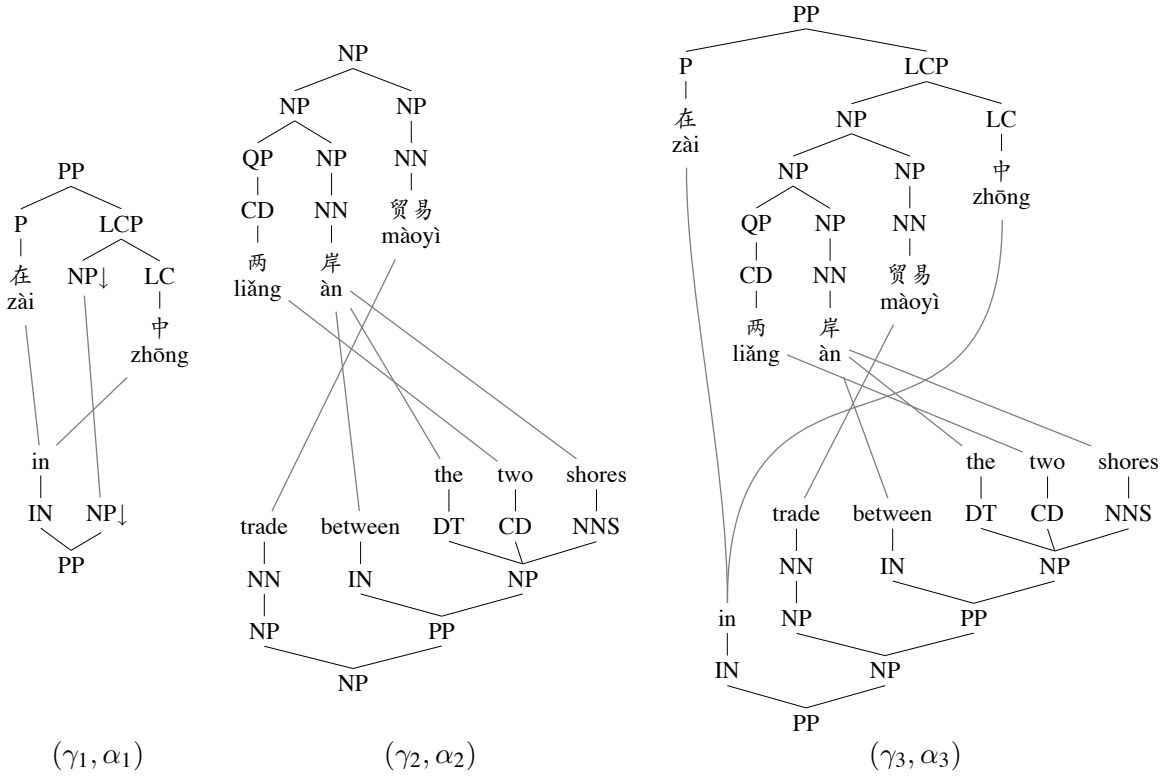


Figure 1: Synchronous tree substitution. Rule (γ_2, α_2) is substituted into rule (γ_1, α_1) to yield (γ_3, α_3).

target-language instead of source-language symbols.

- The substitution nodes of γ are aligned bijectively with those of α.
- The terminal-labeled frontier nodes of γ are aligned (many-to-many) with those of α.

In the *substitution* operation, an aligned pair of substitution nodes is rewritten with an elementary tree pair. The labels of the substitution nodes must match the root labels of the elementary trees with which they are rewritten (but we will relax this constraint below). See Figure 1 for examples of elementary tree pairs and substitution.

2.1 Exact tree-to-tree extraction

The use of STSGs for translation was proposed in the Data-Oriented Parsing literature (Poutsma, 2000; Hearne and Way, 2003) and by Eisner (2003). Both of these proposals are more ambitious about handling spurious ambiguity than approaches derived from phrase-based translation usually have been (the former uses random sampling to sum over equivalent derivations during decoding, and the latter uses dynamic programming to sum over equivalent derivations during training). If we take a more typical approach, which generalizes that of Galley et al. (2004; 2006) and is similar to Stat-XFER (Lavie et al., 2008), we obtain the following grammar extraction method, which we call *exact* tree-to-tree extraction.

	human	automatic
string-to-string	198,445	142,820
max nested	78,361	64,578
tree-to-string	60,939 (78%)	48,235 (75%)
string-to-tree	59,274 (76%)	46,548 (72%)
tree-to-tree	53,084 (68%)	39,049 (60%)

Table 1: Analysis of phrases extracted from Chinese-English newswire data with human and automatic word alignments and parses. As tree constraints are added, the number of phrase pairs drops. Errors in automatic annotations also decrease the number of phrase pairs. Percentages are relative to the maximum number of nested phrase pairs.

Given a pair of source- and target-language parse trees with a word alignment between their leaves, identify all the *phrase pairs* $(\bar{f}, \bar{e})$, i.e., those substring pairs that respect the word align-

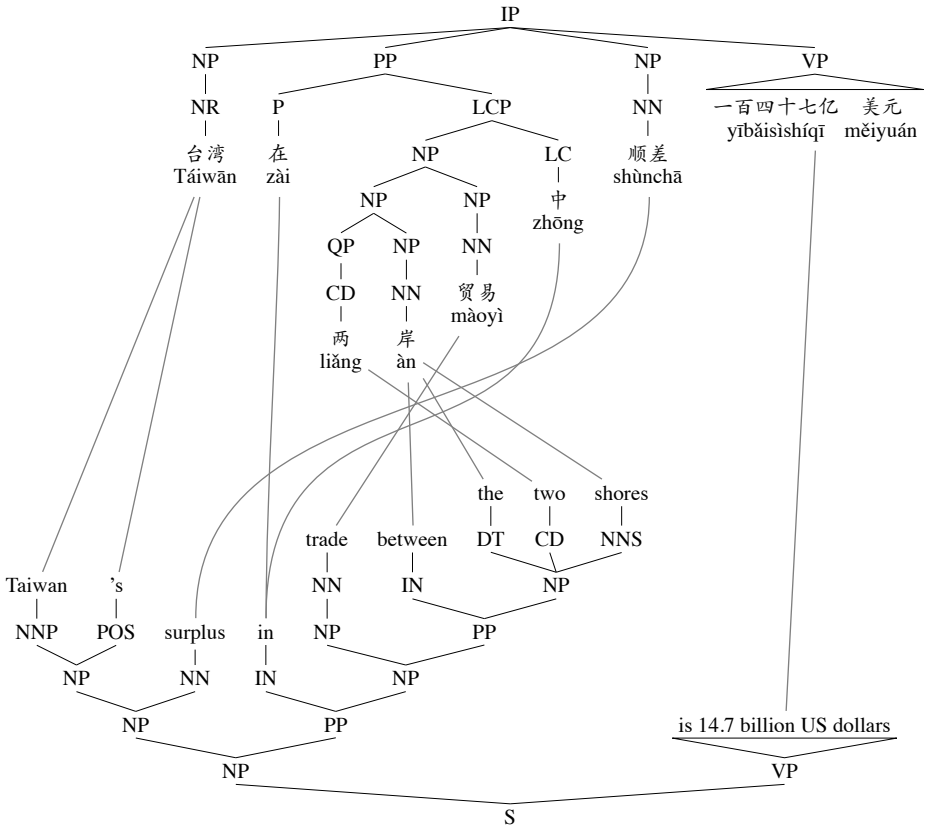


Figure 2: Example Chinese-English sentence pair with human-annotated parse trees and word alignments.

ment in the sense that at least one word in $\bar{f}$ is aligned to a word in $\bar{e}$, and no word in $\bar{f}$ is aligned to a word outside of $\bar{e}$, or vice versa. Then the extracted grammar is the smallest STSG G satisfying:

- If (γ, α) is a pair of subtrees of a training example and the frontiers of γ and α form a phrase pair, then (γ, α) is a rule in G.
- If $(\gamma_2, \alpha_2) \in G$, $(\gamma_3, \alpha_3) \in G$, and (γ_1, α_1) is an elementary tree pair such that substituting (γ_2, α_2) into (γ_1, α_1) results in (γ_3, α_3), then (γ_1, α_1) is a rule in G.

For example, consider the training example in Figure 2, from which the elementary tree pairs shown in Figure 1 can be extracted. The elementary tree pairs (γ_2, α_2) and (γ_3, α_3) are rules in G because their yields are phrase pairs, and (γ_1, α_1) results from subtracting (γ_2, α_2) from (γ_3, α_3).

2.2 Fuzzy tree-to-tree extraction

Exact tree-to-tree translation requires that translation rules deal with syntactic constituents on both the source and target side, which reduces the number of eligible phrases. Table 1 shows an analysis of phrases extracted from human word-aligned and parsed data and automatically word-aligned and parsed data.[1] The first line shows the number of phrase-pair occurrences that are extracted in the absence of syntactic constraints,[2] and the second line shows the maximum number of *nested* phrase-pair occurrences, which is the most that exact syntax-based extraction can achieve. Whereas tree-to-string extraction and string-to-tree extraction permit 70–80% of the maximum possible number of phrase pairs, tree-to-tree extraction only permits 60–70%.

Why does this happen? We can see that moving from human annotations to automatic annotations decreases not only the absolute number of phrase pairs, but the percentage of phrases that pass the syntactic filters. Wellington et al. (2006), in a more systematic study, find that, of sentences where the tree-to-tree constraint blocks rule extraction, the majority are due to parser errors. To address this problem, Liu et al. (2009) extract rules from pairs

[1] The first 2000 sentences from the GALE Phase 4 Chinese Parallel Word Alignment and Tagging Part 1 (LDC2009E83) and the Chinese News Translation Text Part 1 (LDC2005T06), respectively.

[2] Only counting phrases that have no unaligned words at their endpoints.

of packed forests instead of pairs of trees. Since a packed forest is much more likely to include the correct tree, it is less likely that parser errors will cause good rules to be filtered out.

However, even on human-annotated data, tree-to-tree extraction misses many rules, and many such rules would seem to be useful. For example, in Figure 2, the whole English phrase "Taiwan's. . .shores" is an NP, but its Chinese counterpart is not a constituent. Furthermore, neither "surplus. . .shores" nor its Chinese counterpart are constituents. But both rules are arguably useful for translation. Wellington et al. therefore argue that in order to extract as many rules as possible, a more powerful formalism than synchronous CFG/TSG is required: for example, generalized multitext grammar (Melamed et al., 2004), which is equivalent to synchronous set-local multicomponent CFG/TSG (Weir, 1988).

But the problem illustrated in Figure 2 does not reflect a very deep fact about syntax or cross-lingual divergences, but rather choices in annotation style that interact badly with the exact tree-to-tree extraction heuristic. On the Chinese side, the IP is too flat (because 台湾/Táiwān has been analyzed as a topic), whereas the more articulated structure

(1) [NP Táiwān [NP [PP zaì . . .] shùnchā]]

would also be quite reasonable. On the English side, the high attachment of the PP disagrees with the corresponding Chinese structure, but low attachment also seems reasonable:

(2) [NP [NP Taiwan's] [NP surplus in trade. . .]]

Thus even in the gold-standard parse trees, phrase structure can be underspecified (like the flat IP above) or uncertain (like the PP attachment above).

For this reason, some approaches work with a more flexible notion of constituency. *Synchronous tree-sequence–substitution grammar (STSSG)* allows either side of a rule to comprise a sequence of trees instead of a single tree (Zhang et al., 2008). In the substitution operation, a sequence of sister substitution nodes is rewritten with a tree sequence of equal length (see Figure 3a). This extra flexibility effectively makes the analysis (1) available to us.

Any STSSG can be converted into an equivalent STSG via the creation of virtual nodes (see Figure 3b): for every elementary tree sequence with roots $X_1, \ldots, X_n$, create a new root node with a

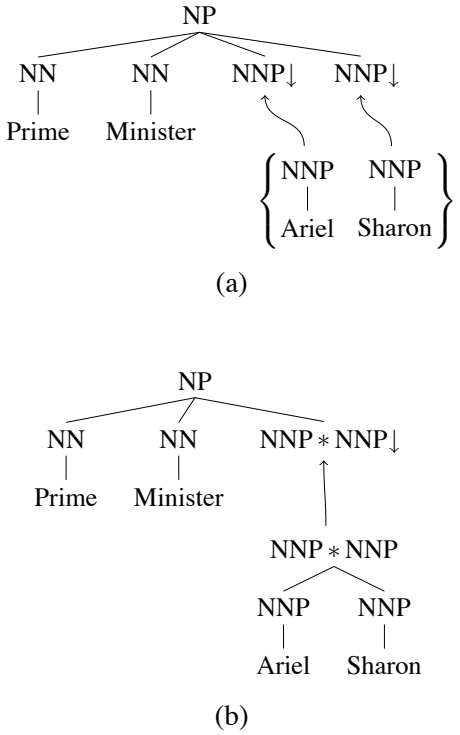


Figure 3: (a) Example tree-sequence substitution grammar and (b) its equivalent SAMT-style tree-substitution grammar.

complex label $X_1 * \cdots * X_n$ immediately dominating the old roots, and replace every sequence of substitution sites $X_1, \ldots, X_n$ with a single substitution site $X_1 * \cdots * X_n$. This is essentially what *syntax-augmented MT (SAMT)* does, in the string-to-tree setting (Zollmann and Venugopal, 2006). In addition, SAMT drops the requirement that the X_i are sisters, and uses categories X / Y (an X missing a Y on the right) and $Y \setminus X$ (an X missing a Y on the left) in the style of categorial grammar (Bar-Hillel, 1953). Under this flexible notion of constituency, both (1) and (2) become available, albeit with more complicated categories.

Both STSSG and SAMT are examples of what we might call *fuzzy tree-to-tree extraction*. We follow this approach here as well: as in STSSG, we work on tree-to-tree data, and we use the complex categories of SAMT. Moreover, we allow the product categories $X_1 * \cdots * X_n$ to be of any length n, and we allow the slash categories to take any number of arguments on either side. Thus every phrase can be assigned a (possibly very complex) syntactic category, so that fuzzy tree-to-tree extraction does not lose any rules relative to string-to-string extraction.

On the other hand, if several rules are extracted

that differ only in their nonterminal labels, only the most-frequent rule is kept, and its count is the total count of all the rules. This means that there is a one-to-one correspondence between the rules extracted by fuzzy tree-to-tree extraction and hierarchical string-to-string extraction.

2.3 Nesting phrases

Fuzzy tree-to-tree extraction (like string-to-string extraction) generates many times more rules than exact tree-to-tree extraction does. In Figure 2, we observed that the flat structure of the Chinese IP prevented exact tree-to-tree extraction from extracting a rule containing just part of the IP, for example:

(3) [PP zaì ...] [NP shùnchā]

(4) [NP Táiwān] [PP zaì ...] [NP shùnchā]

(5) [PP zaì ...] [NP shùnchā] [VP ... měiyuán]

Fuzzy tree-to-tree extraction allows any of these to be the source side of a rule. We might think of it as effectively restructuring the trees by inserting nodes with complex labels. However, it is not possible to represent this restructuring with a single tree (see Figure 4). More formally, let us say that two phrases $w_i \cdots w_{j-1}$ and $w_{i'} \cdots w_{j'-1}$ *nest* if $i \leq i' < j' \leq j$ or $i' \leq i < j < j'$; otherwise, they *cross*. The two Chinese phrases (4) and (5) cross, and therefore cannot both be constituents in the same tree. In other words, exact tree-to-tree extraction commits to a single structural analysis but fuzzy tree-to-tree extraction pursues many restructured analyses at once.

We can strike a compromise by continuing to allow SAMT-style complex categories, but committing to a single analysis by requiring all phrases to nest. To do this, we use a simple heuristic. Iterate through all the phrase pairs $(\bar{f}, \bar{e})$ in the following order:

1. sort by whether $\bar{f}$ and $\bar{e}$ can be assigned a simple syntactic category (both, then one, then neither); if there is a tie,
2. sort by how many syntactic constituents $\bar{f}$ and $\bar{e}$ cross (low to high); if there is a tie,
3. give priority to $(\bar{f}, \bar{e})$ if neither $\bar{f}$ nor $\bar{e}$ begins or ends with punctuation; if there is a tie, finally
4. sort by the position of $\bar{f}$ in the source-side string (right to left).

For each phrase pair, accept it if it does not cross any previously accepted phrase pair; otherwise, reject it.

Because this heuristic produces a set of nesting phrases, we can represent them all in a single restructured tree. In Figure 4, this heuristic chooses structure (a) because the English-side counterpart of IP/VP has the simple category NP.

3 Decoding

In decoding, the rules extracted during training must be reassembled to form a derivation whose source side matches the input sentence. In the exact tree-to-tree approach, whenever substitution is performed, the root labels of the substituted trees must match the labels of the substitution nodes—call this the *matching constraint*. Because this constraint must be satisfied on both the source and target side, it can become difficult to generalize well from training examples to new input sentences.

Venugopal et al. (2009), in the string-to-tree setting, attempt to soften the data-fragmentation effect of the matching constraint: instead of trying to find the single derivation with the highest probability, they sum over derivations that differ only in their nonterminal labels and try to find the single derivation-class with the highest probability. Still, only derivations that satisfy the matching constraint are included in the summation.

But in some cases we may want to soften the matching constraint itself. Some syntactic categories are similar enough to be considered compatible: for example, if a rule rooted in VBD (past-tense verb) could substitute into a site labeled VBZ (present-tense verb), it might still generate correct output. This is all the more true with the addition of SAMT-style categories: for example, if a rule rooted in ADVP $*$ VP could substitute into a site labeled VP, it would very likely generate correct output.

Since we want syntactic information to help the model make good translation choices, not to rule out potentially correct choices, we can change the way the information is used during decoding: we allow any rule to substitute into any site, but let the model learn which substitutions are better than others. To do this, we add the following features to the model:

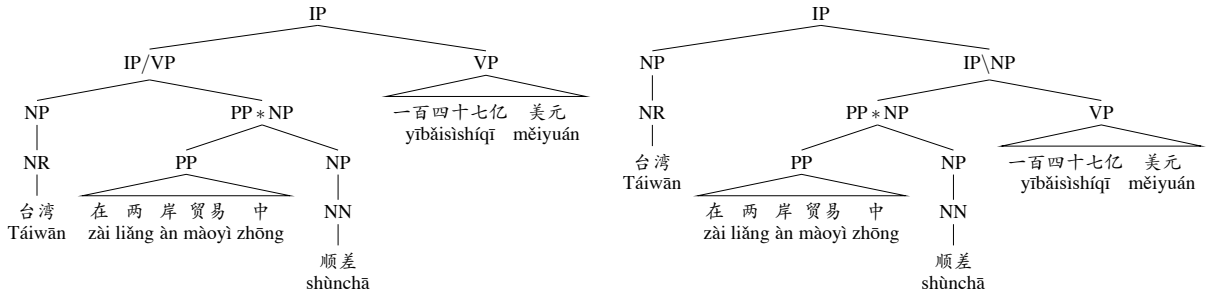


Figure 4: Fuzzy tree-to-tree extraction effectively restructures the Chinese tree from Figure 2 in two ways but does not commit to either one.

- $match^f$ counts the number of substitutions where the label of the source side of the substitution site matches the root label of the source side of the rule, and $\neg match^f$ counts those where the labels do not match.

- $subst^f_{X \rightarrow Y}$ counts the number of substitutions where the label of the source side of the substitution site is X and the root label of the source side of the rule is Y.

- $match^e$, $\neg match^e$, and $subst^e_{X \rightarrow Y}$ do the same for the target side.

- $root_{X,X'}$ counts the number of rules whose root label on the source side is X and whose root label on the target side is X'.[3]

For example, in the derivation of Figure 1, the following features would fire:

$$\begin{aligned} match^f &= 1 \\ subst^f_{\text{NP} \rightarrow \text{NP}} &= 1 \\ match^e &= 1 \\ subst^e_{\text{NP} \rightarrow \text{NP}} &= 1 \\ root_{\text{NP,NP}} &= 1 \end{aligned}$$

The decoding algorithm then operates as in hierarchical phrase-based translation. The decoder has to store in each hypothesis the source and target root labels of the partial derivation, but these labels are used for calculating feature vectors only and not for checking well-formedness of derivations. This additional state does increase the search space of the decoder, but we did not change any pruning settings.

[3]Thanks to Adam Pauls for suggesting this feature class.

4 Experiments

To compare the methods described above with hierarchical string-to-string translation, we ran experiments on both Chinese-English and Arabic-English translation.

4.1 Setup

The sizes of the parallel texts used are shown in Table 2. We word-aligned the Chinese-English parallel text using GIZA++ followed by link deletion (Fossum et al., 2008), and the Arabic-English parallel text using a combination of GIZA++ and LEAF (Fraser and Marcu, 2007). We parsed the source sides of both parallel texts using the Berkeley parser (Petrov et al., 2006), trained on the Chinese Treebank 6 and Arabic Treebank parts 1–3, and the English sides using a reimplementation of the Collins parser (Collins, 1997).

For string-to-string extraction, we used the same constraints as in previous work (Chiang, 2007), with differences shown in Table 2. Rules with nonterminals were extracted from a subset of the data (labeled "Core" in Table 2), and rules without nonterminals were extracted from the full parallel text. Fuzzy tree-to-tree extraction was performed using analogous constraints. For exact tree-to-tree extraction, we used simpler settings: no limit on initial phrase size or unaligned words, and a maximum of 7 frontier nodes on the source side.

All systems used the glue rule (Chiang, 2005), which allows the decoder, working bottom-up, to stop building hierarchical structure and instead concatenate partial translations without any reordering. The model attaches a weight to the glue rule so that it can learn from data whether to build shallow or rich structures, but for efficiency's sake the decoder has a hard limit, called the *distortion*

		Chi-Eng	Ara-Eng
Core	training words	32+38M	28+34M
	initial phrase size	10	15
	final rule size	6	6
	nonterminals	2	2
	loose source	0	∞
	loose target	0	2
Full	training words	240+260M	190+220M
	final rule size	6	6
	nonterminals	0	0
	loose source	∞	∞
	loose target	1	2

Table 2: Rule extraction settings used for experiments. "Loose source/target" is the maximum number of unaligned source/target words at the endpoints of a phrase.

limit, above which the glue rule must be used.

We trained two 5-gram language models: one on the combined English halves of the bitexts, and one on two billion words of English. These were smoothed using modified Kneser-Ney (Chen and Goodman, 1998) and stored using randomized data structures similar to those of Talbot and Brants (2008).

The base feature set for all systems was similar to the expanded set recently used for Hiero (Chiang et al., 2009), but with bigram features (source and target word) instead of trigram features (source and target word and neighboring source word). For all systems but the baselines, the features described in Section 3 were added. The systems were trained using MIRA (Crammer and Singer, 2003; Chiang et al., 2009) on a tuning set of about 3000 sentences of newswire from NIST MT evaluation data and GALE development data, disjoint from the training data. We optimized feature weights on 90% of this and held out the other 10% to determine when to stop.

4.2 Results

Table 3 shows the scores on our development sets and test sets, which are about 3000 and 2000 sentences, respectively, of newswire drawn from NIST MT evaluation data and GALE development data and disjoint from the tuning data.

For Chinese, we first tried increasing the distortion limit from 10 words to 20. This limit controls how deeply nested the tree structures built by the decoder are, and we want to see whether adding syntactic information leads to more complex structures. This change by itself led to an increase in the BLEU score. We then compared against two systems using tree-to-tree grammars. Using exact tree-to-tree extraction, we got a much smaller grammar, but decreased accuracy on all but the Chinese-English test set, where there was no significant change. But with fuzzy tree-to-tree extraction, we obtained an improvement of +0.6 on both Chinese-English sets, and +0.7/+0.8 on the Arabic-English sets.

Applying the heuristic for nesting phrases reduced the grammar sizes dramatically (by a factor of 2.4 for Chinese and 4.2 for Arabic) but, interestingly, had almost no effect on translation quality: a slight decrease in BLEU on the Arabic-English development set and no significant difference on the other sets. This suggests that the strength of fuzzy tree-to-tree extraction lies in its ability to break up flat structures and to reconcile the source and target trees with each other, rather than multiple restructurings of the training trees.

4.3 Rule usage

We then took a closer look at the behavior of the string-to-string and fuzzy tree-to-tree grammars (without the nesting heuristic). Because the rules of these grammars are in one-to-one correspondence, we can analyze the string-to-string system's derivations as though they had syntactic categories. First, Table 4 shows that the system using the tree-to-tree grammar used the glue rule much less and performed more matching substitutions. That is, in order to minimize errors on the tuning set, the model learned to build syntactically richer and more well-formed derivations.

Tables 5 and 6 show how the new syntax features affected particular substitutions. In general we see a shift towards more matching substitutions; correct placement of punctuation is particularly emphasized. Several changes appear to have to do with definiteness of NPs: on the English side, adding the syntax features encourages matching substitutions of type DT \ NP-C (anarthrous NP), but discourages DT \ NP-C and NN from substituting into NP-C and vice versa. For example, a translation with the rewriting NP-C $\rightarrow$ DT \ NP-C begins with "24th meeting of the Standing Committee. . .," but the system using the fuzzy tree-to-tree grammar changes this to "The 24th meeting of the Standing Committee. . . ."

The *root* features had a less noticeable effect on

task	extraction	dist. lim.	rules	features	BLEU dev	BLEU test
Chi-Eng	string-to-string	10	440M	1k	32.7	23.4
	string-to-string	20	440M	1k	33.3	23.7
	tree-to-tree exact	20	50M	5k	32.8	23.9
	tree-to-tree fuzzy	20	440M	160k	33.9	24.3
	+ nesting	20	180M	79k	33.9	24.3
Ara-Eng	string-to-string	10	790M	1k	48.7	48.9
	tree-to-tree exact	10	38M	5k	46.6	47.5
	tree-to-tree fuzzy	10	790M	130k	49.4	49.7
	+ nesting	10	190M	66k	49.2	49.8

Table 3: On both the Chinese-English and Arabic-English translation tasks, fuzzy tree-to-tree extraction outperforms exact tree-to-tree extraction and string-to-string extraction. Brackets indicate statistically insignificant differences ($p \geq 0.05$).

rule choice; one interesting change was that the frequency of rules with Chinese root VP / IP and English root VP / S-C increased from 0.2% to 0.7%: apparently the model learned that it is good to use rules that pair Chinese and English verbs that subcategorize for sentential complements.

5 Conclusion

Though exact tree-to-tree translation tends to hamper translation quality by imposing too many constraints during both grammar extraction and decoding, we have shown that using both source and target syntax improves translation accuracy when the model is given the opportunity to learn from data how strongly to apply syntactic constraints. Indeed, we have found that the model learns on its own to choose syntactically richer and more well-formed structures, demonstrating that source- and target-side syntax can be used together profitably as long as they are not allowed to overconstrain the translation model.

Acknowledgements

Thanks to Steve DeNeefe, Adam Lopez, Jonathan May, Miles Osborne, Adam Pauls, Richard Schwartz, and the anonymous reviewers for their valuable help. This research was supported in part by DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies and DARPA contract HR0011-09-1-0028. *S. D. G.*

task	side	kind	frequency (%) s-to-s	frequency (%) t-to-t
Chi-Eng	source	glue	25	18
		match	17	30
		mismatch	58	52
	target	glue	25	18
		match	9	23
		mismatch	66	58
Ara-Eng	source	glue	36	19
		match	17	34
		mismatch	48	47
	target	glue	36	19
		match	11	29
		mismatch	53	52

Table 4: Moving from string-to-string (s-to-s) extraction to fuzzy tree-to-tree (t-to-t) extraction decreases glue rule usage and increases the frequency of matching substitutions.

kind	frequency (%) s-to-s	t-to-t
NP → NP	16.0	20.7
VP → VP	3.3	5.9
NN → NP	3.1	1.3
NP → VP	2.5	0.8
NP → NN	2.0	1.4
NP → entity	1.4	1.6
NN → NN	1.1	1.0
QP → entity	1.0	1.3
VV → VP	1.0	0.7
PU → NP	0.8	1.1
VV → VP ∗ PU	0.2	1.2
PU → PU	0.1	3.8

Table 5: Comparison of frequency of source-side rewrites in Chinese-English translation between string-to-string (s-to-s) and fuzzy tree-to-tree (t-to-t) grammars. All rewrites occurring more than 1% of the time in either system are shown. The label "entity" stands for handwritten rules for named entities and numbers.

kind	frequency (%) s-to-s	t-to-t
NP-C → NP-C	5.3	8.7
NN → NN	1.7	3.0
NP-C → entity	1.1	1.4
DT \ NP-C → DT \ NP-C	1.1	2.6
NN → NP-C	0.8	0.4
NP-C → VP	0.8	1.1
DT \ NP-C → NP-C	0.8	0.5
NP-C → DT \ NP-C	0.6	0.4
JJ → JJ	0.5	1.8
NP-C → NN	0.5	0.3
PP → PP	0.4	1.7
VP-C → VP-C	0.4	1.2
VP → VP	0.4	1.4
IN → IN	0.1	1.8
, → ,	0.1	1.7

Table 6: Comparison of frequency of target-side rewrites in Chinese-English translation between string-to-string (s-to-s) and fuzzy tree-to-tree (t-to-t) grammars. All rewrites occurring more than 1% of the time in either system are shown, plus a few more of interest. The label "entity" stands for handwritten rules for named entities and numbers.

References

Vamshi Ambati and Alon Lavie. 2008. Improving syntax driven translation models by re-structuring divergent and non-isomorphic parse tree structures. In *Proc. AMTA-2008 Student Research Workshop*, pages 235–244.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29(1):47–58.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang, Wei Wang, and Kevin Knight. 2009. 11,001 new features for statistical machine translation. In *Proc. NAACL HLT 2009*, pages 218–226.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL 2005*, pages 263–270.

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

Michael Collins. 1997. Three generative lexicalised models for statistical parsing. In *Proc. ACL-EACL*, pages 16–23.

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

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. ACL 2003 Companion Volume*, pages 205–208.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment for syntax-based statistical machine translation. In *Proc. Third Workshop on Statistical Machine Translation*, pages 44–52.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proc. EMNLP 2007*, pages 51–60.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL 2004*, pages 273–280.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. COLING-ACL 2006*, pages 961–968.

Mary Hearne and Andy Way. 2003. Seeing the wood for the trees: Data-Oriented Translation. In *Proc. MT Summit IX*, pages 165–172.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA 2006*, pages 65–73.

Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proc. SSST-2*, pages 87–95.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. COLING-ACL 2006*, pages 609–616.

Yang Liu, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. ACL 2009*, pages 558–566.

I. Dan Melamed, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *Proc. ACL 2004*, pages 661–668.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING-ACL 2006*, pages 433–440.

Arjen Poutsma. 2000. Data-Oriented Translation. In *Proc. COLING 2000*, pages 635–641.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proc. ACL-08: HLT*, pages 505–513.

Ashish Venugopal, Andreas Zollmann, Noah A. Smith, and Stephan Vogel. 2009. Preference grammars: Softening syntactic constraints to improve statistical machine translation. In *Proc. NAACL HLT 2009*, pages 236–244.

David J. Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proc. COLING-ACL 2006*, pages 977–984.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. ACL-08: HLT*, pages 559–567.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. Workshop on Statistical Machine Translation*, pages 138–141.

Discriminative Modeling of Extraction Sets for Machine Translation

John DeNero and **Dan Klein**
Computer Science Division
University of California, Berkeley
{denero,klein}@cs.berkeley.edu

Abstract

We present a discriminative model that directly predicts which set of phrasal translation rules should be extracted from a sentence pair. Our model scores *extraction sets*: nested collections of all the overlapping phrase pairs consistent with an underlying word alignment. Extraction set models provide two principle advantages over word-factored alignment models. First, we can incorporate features on phrase pairs, in addition to word links. Second, we can optimize for an extraction-based loss function that relates directly to the end task of generating translations. Our model gives improvements in alignment quality relative to state-of-the-art unsupervised and supervised baselines, as well as providing up to a 1.4 improvement in BLEU score in Chinese-to-English translation experiments.

1 Introduction

In the last decade, the field of statistical machine translation has shifted from generating sentences word by word to systems that recycle whole fragments of training examples, expressed as *translation rules*. This general paradigm was first pursued using contiguous phrases (Och et al., 1999; Koehn et al., 2003), and has since been generalized to a wide variety of hierarchical and syntactic formalisms. The training stage of statistical systems focuses primarily on discovering translation rules in parallel corpora.

Most systems discover translation rules via a two-stage pipeline: a parallel corpus is aligned at the word level, and then a second procedure extracts fragment-level rules from word-aligned sentence pairs. This paper offers a model-based alternative to phrasal rule extraction, which merges this two-stage pipeline into a single step. We present a discriminative model that directly predicts which set of phrasal translation rules should be extracted from a sentence pair. Our model predicts *extraction sets*: combinatorial objects that include the set of all overlapping phrasal translation rules consistent with an underlying word-level alignment. This approach provides additional discriminative power relative to word aligners because extraction sets are scored based on the phrasal rules they contain in addition to word-to-word alignment links. Moreover, the structure of our model directly reflects the purpose of alignment models in general, which is to discover translation rules.

We address several challenges to training and applying an extraction set model. First, we would like to leverage existing word-level alignment resources. To do so, we define a deterministic mapping from word alignments to extraction sets, inspired by existing extraction procedures. In our mapping, *possible* alignment links have a precise interpretation that dictates what phrasal translation rules can be extracted from a sentence pair. This mapping allows us to train with existing annotated data sets and use the predictions from word-level aligners as features in our extraction set model.

Second, our model solves a structured prediction problem, and the choice of loss function during training affects model performance. We optimize for a phrase-level F-measure in order to focus learning on the task of predicting phrasal rules rather than word alignment links.

Third, our discriminative approach requires that we perform inference in the space of extraction sets. Our model does not factor over disjoint word-to-word links or minimal phrase pairs, and so existing inference procedures do not directly apply. However, we show that the dynamic program for a block ITG aligner can be augmented to score extraction sets that are indexed by underlying ITG word alignments (Wu, 1997). We also describe a

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1453–1463,
Uppsala, Sweden, 11-16 July 2010.

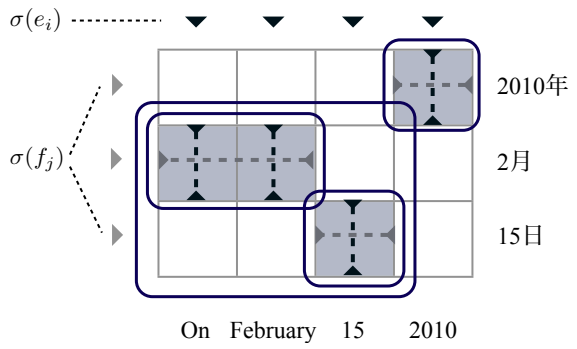


Figure 1: A word alignment $\mathcal{A}$ (shaded grid cells) defines projections $\sigma(e_i)$ and $\sigma(f_j)$, shown as dotted lines for each word in each sentence. The extraction set $R_3(\mathcal{A})$ includes all bispans licensed by these projections, shown as rounded rectangles.

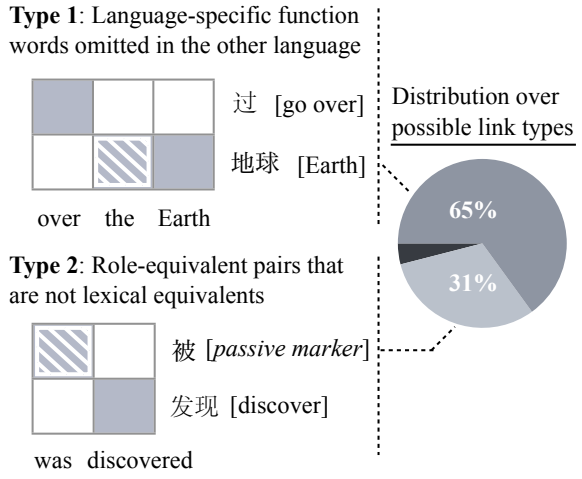


Figure 2: Examples of two types of possible alignment links (striped). These types account for 96% of the possible alignment links in our data set.

coarse-to-fine inference approach that allows us to scale our method to long sentences.

Our extraction set model outperforms both unsupervised and supervised word aligners at predicting word alignments and extraction sets. We also demonstrate that extraction sets are useful for end-to-end machine translation. Our model improves translation quality relative to state-of-the-art Chinese-to-English baselines across two publicly available systems, providing total BLEU improvements of 1.2 in Moses, a phrase-based system, and 1.4 in a Joshua, a hierarchical system (Koehn et al., 2007; Li et al., 2009)

2 Extraction Set Models

The input to our model is an unaligned sentence pair, and the output is an extraction set of phrasal translation rules. Word-level alignments are generated as a byproduct of inference. We first specify the relationship between word alignments and extraction sets, then define our model.

2.1 Extraction Sets from Word Alignments

Rule extraction is a standard concept in machine translation: word alignment constellations license particular sets of overlapping rules, from which subsets are selected according to limits on phrase length (Koehn et al., 2003), number of gaps (Chiang, 2007), count of internal tree nodes (Galley et al., 2006), etc. In this paper, we focus on phrasal rule extraction (i.e., phrase pair extraction), upon which most other extraction procedures are based.

Given a sentence pair $(\mathbf{e}, \mathbf{f})$, phrasal rule extraction defines a mapping from a set of word-to-word alignment links $\mathcal{A} = \{(i, j)\}$ to an *extraction set* of bispans $R_n(\mathcal{A}) = \{[g, h) \Leftrightarrow [k, \ell)\}$, where each bispan links target span $[g, h)$ to source span $[k, \ell)$.[1] The maximum phrase length n ensures that $\max(h - g, \ell - k) \leq n$.

We can describe this mapping via word-to-phrase projections, as illustrated in Figure 1. Let word e_i project to the phrasal span $\sigma(e_i)$, where

$$\sigma(e_i) = \left[\min_{j \in J_i} j \ , \ \max_{j \in J_i} j + 1 \right) \qquad (1)$$
$$J_i = \{j : (i, j) \in \mathcal{A}\}$$

and likewise each word f_j projects to a span of $\mathbf{e}$. Then, $R_n(\mathcal{A})$ includes a bispan $[g, h) \Leftrightarrow [k, \ell)$ iff

$$\sigma(e_i) \subseteq [k, \ell) \qquad \forall i \in [g, h)$$
$$\sigma(f_j) \subseteq [g, h) \qquad \forall j \in [k, \ell)$$

That is, every word in one of the phrasal spans must project within the other. This mapping is deterministic, and so we can interpret a word-level alignment $\mathcal{A}$ as also specifying the phrasal rules that should be extracted from a sentence pair.

2.2 Possible and Null Alignment Links

We have not yet accounted for two special cases in annotated corpora: *possible* alignments and *null* alignments. To analyze these annotations, we consider a particular data set: a hand-aligned portion

[1] We use the fencepost indexing scheme used commonly for parsing. Words are 0-indexed. Spans are inclusive on the lower bound and exclusive on the upper bound. For example, the span $[0, 2)$ includes the first two words of a sentence.

of the NIST MT02 Chinese-to-English test set, which has been used in previous alignment experiments (Ayan et al., 2005; DeNero and Klein, 2007; Haghighi et al., 2009).

Possible links account for 22% of all alignment links in these data, and we found that most of these links fall into two categories. First, possible links are used to align function words that have no equivalent in the other language, but colocate with aligned content words, such as English determiners. Second, they are used to mark pairs of words or short phrases that are not lexical equivalents, but which play equivalent roles in each sentence. Figure 2 shows examples of these two use cases, along with their corpus frequencies.[2]

On the other hand, null alignments are used sparingly in our annotated data. More than 90% of words participate in some alignment link. The unaligned words typically express content in one sentence that is absent in its translation.

Figure 3 illustrates how we interpret possible and null links in our projection. Possible links are typically not included in extraction procedures because most aligners predict only sure links. However, we see a natural interpretation for possible links in rule extraction: they license phrasal rules that both include and exclude them. We exclude null alignments from extracted phrases because they often indicate a mismatch in content.

We achieve these effects by redefining the projection operator σ. Let $\mathcal{A}^{(s)}$ be the subset of $\mathcal{A}$ that are *sure* links, then let the index set J_i used for projection σ in Equation 1 be

$$J_i = \begin{cases} \{j : (i,j) \in \mathcal{A}^{(s)}\} & \text{if } \exists j : (i,j) \in \mathcal{A}^{(s)} \\ \{-1, |\mathbf{f}|\} & \text{if } \nexists j : (i,j) \in \mathcal{A} \\ \{j : (i,j) \in \mathcal{A}\} & \text{otherwise} \end{cases}$$

Here, J_i is a set of integers, and $\sigma(e_i)$ for null aligned e_i will be $[-1, |\mathbf{f}| + 1)$ by Equation 1.

Of course, the characteristics of our aligned corpus may not hold for other annotated corpora or other language pairs. However, we hope that the overall effectiveness of our modeling approach will influence future annotation efforts to build corpora that are consistent with this interpretation.

2.3 A Linear Model of Extraction Sets

We now define a linear model that scores extraction sets. We restrict our model to score only *coherent* extraction sets $R_n(\mathcal{A})$, those that are licensed by an underlying word alignment $\mathcal{A}$ with sure alignments $\mathcal{A}^{(s)} \subseteq \mathcal{A}$. Conditioned on a sentence pair $(\mathbf{e}, \mathbf{f})$ and maximum phrase length n, we score extraction sets via a feature vector $\phi(\mathcal{A}^{(s)}, R_n(\mathcal{A}))$ that includes features on sure links $(i,j) \in \mathcal{A}^{(s)}$ and features on the bispans in $R_n(\mathcal{A})$ that link $[g,h)$ in $\mathbf{e}$ to $[k,\ell)$ in $\mathbf{f}$:

$$\phi(\mathcal{A}^{(s)}, R_n(\mathcal{A})) = \sum_{(i,j)\in\mathcal{A}^{(s)}} \phi_a(i,j) + \sum_{[g,h)\Leftrightarrow[k,\ell)\in R_n(\mathcal{A})} \phi_b(g,h,k,\ell)$$

Because the projection operator $R_n(\cdot)$ is a deterministic function, we can abbreviate $\phi(\mathcal{A}^{(s)}, R_n(\mathcal{A}))$ as $\phi(\mathcal{A})$ without loss of information, although we emphasize that $\mathcal{A}$ is a set of sure and possible alignments, and $\phi(\mathcal{A})$ does not decompose as a sum of vectors on individual word-level alignment links. Our model is parameterized by a weight vector θ, which scores an extraction set $R_n(\mathcal{A})$ as $\theta \cdot \phi(\mathcal{A})$.

To further limit the space of extraction sets we are willing to consider, we restrict $\mathcal{A}$ to block inverse transduction grammar (ITG) alignments, a space that allows many-to-many alignments through phrasal terminal productions, but otherwise enforces at-most-one-to-one phrase matchings with ITG reordering patterns (Cherry and Lin, 2007; Zhang et al., 2008). The ITG constraint

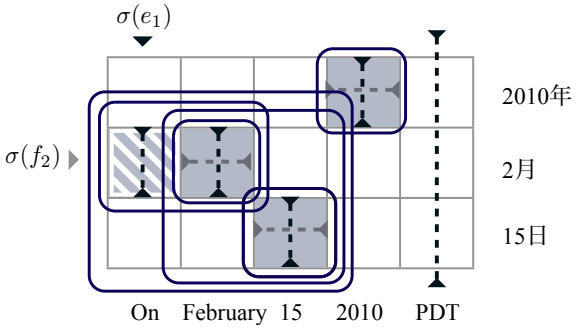


Figure 3: Possible links constrain the word-to-phrase projection of otherwise unaligned words, which in turn license overlapping phrases. In this example, $\sigma(f_2) = [1,2)$ does not include the possible link at $(1,0)$ because of the sure link at $(1,1)$, but $\sigma(e_1) = [1,2)$ does use the possible link because it would otherwise be unaligned. The word "PDT" is null aligned, and so its projection $\sigma(e_4) = [-1,4)$ extends beyond the bounds of the sentence, excluding "PDT" from all phrase pairs.

[2]We collected corpus frequencies of possible alignment link types ourselves on a sample of the hand-aligned data set.

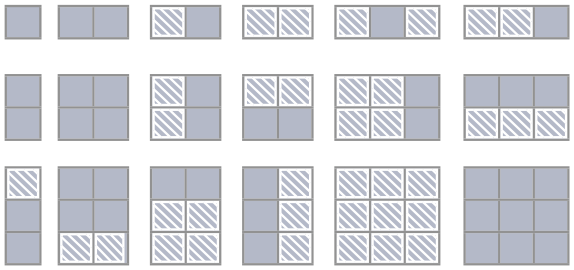


Figure 4: Above, we show a representative subset of the block alignment patterns that serve as terminal productions of the ITG that restricts the output space of our model. These terminal productions cover up to $n = 3$ words in each sentence and include a mixture of sure (filled) and possible (striped) word-level alignment links.

is more computationally convenient than arbitrarily ordered phrase matchings (Wu, 1997; DeNero and Klein, 2008). However, the space of block ITG alignments is expressive enough to include the vast majority of patterns observed in hand-annotated parallel corpora (Haghighi et al., 2009).

In summary, our model scores all $R_n(\mathcal{A})$ for $\mathcal{A} \in \text{ITG}(\mathbf{e}, \mathbf{f})$ where $\mathcal{A}$ can include block terminals of size up to n. In our experiments, $n = 3$. Unlike previous work, we allow possible alignment links to appear in the block terminals, as depicted in Figure 4.

3 Model Estimation

We estimate the weights θ of our extraction set model discriminatively using the margin-infused relaxed algorithm (MIRA) of Crammer and Singer (2003)—a large-margin, perceptron-style, online learning algorithm. MIRA has been used successfully in MT to estimate both alignment models (Haghighi et al., 2009) and translation models (Chiang et al., 2008).

For each training example, MIRA requires that we find the alignment $\mathcal{A}_m$ corresponding to the highest scoring extraction set $R_n(\mathcal{A}_m)$ under the current model,

$$\mathcal{A}_m = \arg\max_{\mathcal{A} \in \text{ITG}(\mathbf{e},\mathbf{f})} \theta \cdot \phi(\mathcal{A}) \qquad (2)$$

Section 4 describes our approach to solving this search problem for model inference.

MIRA updates away from $R_n(A_m)$ and toward a gold extraction set $R_n(A_g)$. Some hand-annotated alignments are outside of the block ITG model class. Hence, we update toward the extraction set for a pseudo-gold alignment $\mathcal{A}_g \in \text{ITG}(\mathbf{e}, \mathbf{f})$ with minimal distance from the true reference alignment $\mathcal{A}_t$.

$$\mathcal{A}_g = \arg\min_{\mathcal{A} \in \text{ITG}(\mathbf{e},\mathbf{f})} |\mathcal{A} \cup \mathcal{A}_t - \mathcal{A} \cap \mathcal{A}_t| \quad (3)$$

Inference details appear in Section 4.3.

Given $\mathcal{A}_g$ and $\mathcal{A}_m$, we update the model parameters away from $\mathcal{A}_m$ and toward $\mathcal{A}_g$.

$$\theta \leftarrow \theta + \tau \cdot (\phi(\mathcal{A}_g) - \phi(\mathcal{A}_m))$$

where τ is the minimal step size that will ensure we prefer $\mathcal{A}_g$ to $\mathcal{A}_m$ by a margin greater than the loss $L(\mathcal{A}_m; \mathcal{A}_g)$, capped at some maximum update size C to provide regularization. We use $C = 0.01$ in experiments. The step size is a closed form function of the loss and feature vectors: $\tau =$

$$\min\left(C, \frac{L(\mathcal{A}_m; \mathcal{A}_g) - \theta \cdot (\phi(\mathcal{A}_g) - \phi(\mathcal{A}_m))}{||\phi(\mathcal{A}_g) - \phi(\mathcal{A}_m)||_2^2}\right)$$

We train the model for 30 iterations over the training set, shuffling the order each time, and we average the weight vectors observed after each iteration to estimate our final model.

3.1 Extraction Set Loss Function

In order to focus learning on predicting the right bispans, we use an extraction-level loss $L(\mathcal{A}_m; \mathcal{A}_g)$: an F-measure of the overlap between bispans in $R_n(\mathcal{A}_m)$ and $R_n(\mathcal{A}_g)$. This measure has been proposed previously to evaluate alignment systems (Ayan and Dorr, 2006). Based on preliminary translation results during development, we chose bispan F_5 as our loss:

$$\text{Pr}(\mathcal{A}_m) = |R_n(\mathcal{A}_m) \cap R_n(\mathcal{A}_g)|/|R_n(\mathcal{A}_m)|$$
$$\text{Rc}(\mathcal{A}_m) = |R_n(\mathcal{A}_m) \cap R_n(\mathcal{A}_g)|/|R_n(\mathcal{A}_g)|$$
$$\text{F}_5(\mathcal{A}_m; \mathcal{A}_g) = \frac{(1+5^2) \cdot \text{Pr}(\mathcal{A}_m) \cdot \text{Rc}(\mathcal{A}_m)}{5^2 \cdot \text{Pr}(\mathcal{A}_m) + \text{Rc}(\mathcal{A}_m)}$$
$$L(\mathcal{A}_m; \mathcal{A}_g) = 1 - \text{F}_5(\mathcal{A}_m; \mathcal{A}_g)$$

F_5 favors recall over precision. Previous alignment work has shown improvements from adjusting the F-measure parameter (Fraser and Marcu, 2006). In particular, Lacoste-Julien et al. (2006) also chose a recall-biased objective.

Optimizing for a bispan F-measure penalizes alignment mistakes in proportion to their rule extraction consequences. That is, adding a word link that prevents the extraction of many correct phrasal rules, or which licenses many incorrect rules, is strongly discouraged by this loss.

3.2 Features on Extraction Sets

The discriminative power of our model is driven by the features on sure word alignment links $\phi_a(i, j)$ and bispans $\phi_b(g, h, k, \ell)$. In both cases, the most important features come from the predictions of unsupervised models trained on large parallel corpora, which provide frequency and co-occurrence information.

To score word-to-word links, we use the posterior predictions of a jointly trained HMM alignment model (Liang et al., 2006). The remaining features include a dictionary feature, an identical word feature, an absolute position distortion feature, and features for numbers and punctuation.

To score phrasal translation rules in an extraction set, we use a mixture of feature types. Extraction set models allow us to incorporate the same phrasal relative frequency statistics that drive phrase-based translation performance (Koehn et al., 2003). To implement these frequency features, we extract a phrase table from the alignment predictions of a jointly trained unsupervised HMM model using Moses (Koehn et al., 2007), and score bispans using the resulting features. We also include indicator features on lexical templates for the 50 most common words in each language, as in Haghighi et al. (2009). We include indicators for the number of words and Chinese characters in rules. One useful indicator feature exploits the fact that capitalized terms in English tend to align to Chinese words with three or more characters. On 1-by-n or n-by-1 phrasal rules, we include indicator features of fertility for common words.[3]

We also include monolingual phrase features that expose useful information to the model. For instance, English bigrams beginning with "the" are often extractable phrases. English trigrams with a hyphen as the second word are typically extractable, meaning that the first and third words align to consecutive Chinese words. When any conjugation of the word "to be" is followed by a verb, indicating passive voice or progressive tense, the two words tend to align together.

Our feature set also includes bias features on phrasal rules and links, which control the number of null-aligned words and number of rules licensed. In total, our final model includes 4,249 individual features, dominated by various instantiations of lexical templates.

[3]Limiting lexicalized features to common words helps prevent overfitting.

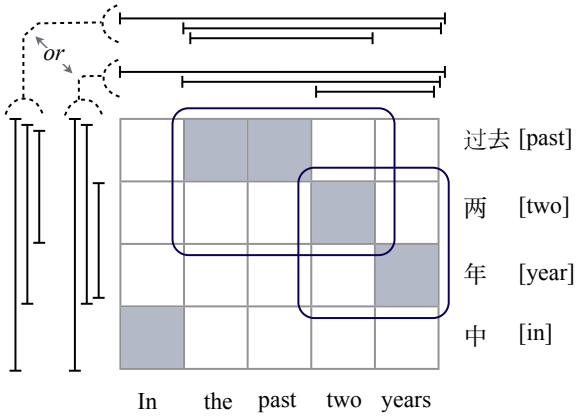


Figure 5: Both possible ITG decompositions of this example alignment will split one of the two highlighted bispans across constituents.

4 Model Inference

Equation 2 asks for the highest scoring extraction set under our model, $R_n(\mathcal{A}_m)$, which we also require at test time. Although we have restricted $\mathcal{A}_m \in \text{ITG}(\mathbf{e}, \mathbf{f})$, our extraction set model does not factor over ITG productions, and so the dynamic program for a vanilla block ITG will not suffice to find $R_n(\mathcal{A}_m)$. To see this, consider the extraction set in Figure 5. An ITG decomposition of the underlying alignment imposes a hierarchical bracketing on each sentence, and some bispan in the extraction set for this alignment will cross any such bracketing. Hence, the score of some licensed bispan will be non-local to the ITG decomposition.

4.1 A Dynamic Program for Extraction Sets

If we treat the maximum phrase length n as a fixed constant, then we can define a dynamic program to search the space of extraction sets. An ITG derivation for some alignment $\mathcal{A}$ decomposes into two sub-derivations for $\mathcal{A}_L$ and $\mathcal{A}_R$.[4] The model score of $\mathcal{A}$, which scores extraction set $R_n(\mathcal{A})$, decomposes over $\mathcal{A}_L$ and $\mathcal{A}_R$, along with any phrasal bispans licensed by adjoining $\mathcal{A}_L$ and $\mathcal{A}_R$.

$$\theta \cdot \phi(\mathcal{A}) = \theta \cdot \phi(\mathcal{A}_L) + \theta \cdot \phi(\mathcal{A}_R) + I(\mathcal{A}_L, \mathcal{A}_R)$$

where $I(\mathcal{A}_L, \mathcal{A}_R)$ is $\theta \cdot \sum \phi(g, h, k, l)$ summed over licensed bispans $[g, h) \Leftrightarrow [k, \ell)$ that overlap the boundary between $\mathcal{A}_L$ and $\mathcal{A}_R$.[5]

[4]We abuse notation in conflating an alignment $\mathcal{A}$ with its derivation. All derivations of the same alignment receive the same score, and we only compute the max, not the sum.

[5]We focus on the case of adjoining two aligned bispans. Our algorithm easily extends to include null alignments, but we focus on the non-null setting for simplicity.

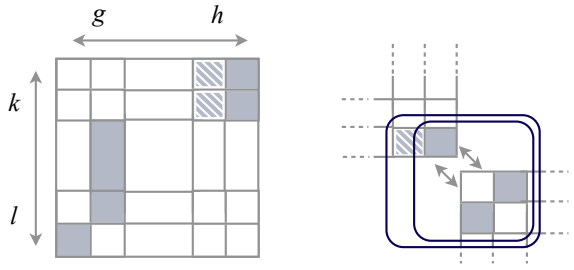


Figure 6: Augmenting the ITG grammar states with the alignment configuration in an $n-1$ deep perimeter of the bispan allows us to score all overlapping phrasal rules introduced by adjoining two bispans. The state must encode whether a sure link appears in each edge column or row, but the specific location of edge links is not required.

In order to compute $I(\mathcal{A}_L, \mathcal{A}_R)$, we need certain information about the alignment configurations of $\mathcal{A}_L$ and $\mathcal{A}_R$ where they adjoin at a corner. The state must represent (a) the specific alignment links in the $n-1$ deep corner of each $\mathcal{A}$, and (b) whether any sure alignments appear in the rows or columns extending from those corners.[6] With this information, we can infer the bispans licensed by adjoining $\mathcal{A}_L$ and $\mathcal{A}_R$, as in Figure 6.

Applying our score recurrence yields a polynomial-time dynamic program. This dynamic program is an instance of ITG bitext parsing, where the grammar uses symbols to encode the alignment contexts described above. This context-as-symbol augmentation of the grammar is similar in character to augmenting symbols with lexical items to score language models during hierarchical decoding (Chiang, 2007).

4.2 Coarse-to-Fine Inference and Pruning

Exhaustive inference under an ITG requires $O(k^6)$ time in sentence length k, and is prohibitively slow when there is no sparsity in the grammar. Maintaining the context necessary to score non-local bispans further increases running time. That is, ITG inference is organized around search states associated with a grammar symbol and a bispan; augmenting grammar symbols also augments this state space.

To parse quickly, we prune away search states using predictions from the more efficient HMM alignment model (Ney and Vogel, 1996). We discard all states corresponding to bispans that are incompatible with 3 or more alignment links under an intersected HMM—a proven approach to pruning the space of ITG alignments (Zhang and Gildea, 2006; Haghighi et al., 2009). Pruning in this way reduces the search space dramatically, but only rarely prohibits correct alignments. The oracle alignment error rate for the block ITG model class is 1.4%; the oracle alignment error rate for this pruned subset of ITG is 2.0%.

To take advantage of the sparsity that results from pruning, we use an agenda-based parser that orders search states from small to large, where we define the size of a bispan as the total number of words contained within it. For each size, we maintain a separate agenda. Only when the agenda for size k is exhausted does the parser proceed to process the agenda for size $k+1$.

We also employ coarse-to-fine search to speed up inference (Charniak and Caraballo, 1998). In the coarse pass, we search over the space of ITG alignments, but score only features on alignment links and bispans that are local to terminal blocks. This simplification eliminates the need to augment grammar symbols, and so we can exhaustively explore the (pruned) space. We then compute outside scores for bispans under a max-sum semiring (Goodman, 1996). In the fine pass with the full extraction set model, we impose a maximum size of 10,000 for each agenda. We order states on agendas by the sum of their inside score under the full model and the outside score computed in the coarse pass, pruning all states not within the fixed agenda beam size.

Search states that are popped off agendas are indexed by their corner locations for fast look-up when constructing new states. For each corner and size combination, built states are maintained in sorted order according to their inside score. This ordering allows us to stop combining states early when the results are falling off the agenda beams. Similar search and beaming strategies appear in many decoders for machine translation (Huang and Chiang, 2007; Koehn and Haddow, 2009; Moore and Quirk, 2007).

4.3 Finding Pseudo-Gold ITG Alignments

Equation 3 asks for the block ITG alignment $\mathcal{A}_g$ that is closest to a reference alignment $\mathcal{A}_t$, which may not lie in ITG(**e**,**f**). We search for

[6] The number of configuration states does not depend on the size of $\mathcal{A}$ because corners have fixed size, and because the position of links within rows or columns is not needed.

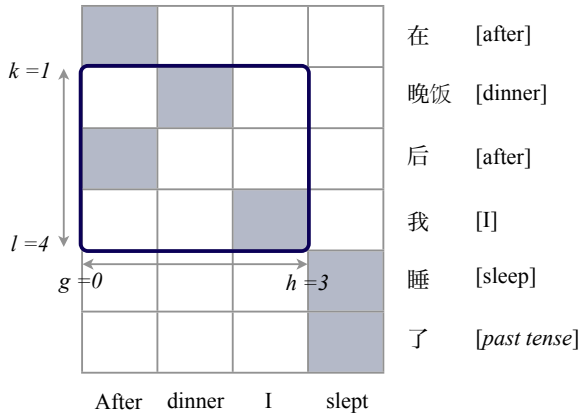


Figure 7: A* search for pseudo-gold ITG alignments uses an admissible heuristic for bispans that counts the number of gold links outside of $[k, \ell)$ but within $[g, h)$. Above, the heuristic is 1, which is also the minimal number of alignment errors that an ITG alignment will incur using this bispan.

$\mathcal{A}_g$ using A* bitext parsing (Klein and Manning, 2003). Search states, which correspond to bispans $[g, h) \Leftrightarrow [k, \ell)$, are scored by the number of errors within the bispan plus the number of $(i, j) \in A_t$ such that $j \in [k, \ell)$ but $i \notin [g, h)$ (recall errors). As an admissible heuristic for the future cost of a bispan $[g, h) \Leftrightarrow [k, \ell)$, we count the number of $(i, j) \in \mathcal{A}_t$ such that $i \in [g, h)$ but $j \notin [k, \ell)$, as depicted in Figure 7. These links will become recall errors eventually. A* search with this heuristic makes no errors, and the time required to compute pseudo-gold alignments is negligible.

5 Relationship to Previous Work

Our model is certainly not the first alignment approach to include structures larger than words. Model-based phrase-to-phrase alignment was proposed early in the history of phrase-based translation as a method for training translation models (Marcu and Wong, 2002). A variety of unsupervised models refined this initial work with priors (DeNero et al., 2008; Blunsom et al., 2009) and inference constraints (DeNero et al., 2006; Birch et al., 2006; Cherry and Lin, 2007; Zhang et al., 2008). These models fundamentally differ from ours in that they stipulate a segmentation of the sentence pair into phrases, and only align the minimal phrases in that segmentation. Our model scores the larger overlapping phrases that result from composing these minimal phrases.

Discriminative alignment is also a well-explored area. Most work has focused on predicting word alignments via partial matching inference algorithms (Melamed, 2000; Taskar et al., 2005; Moore, 2005; Lacoste-Julien et al., 2006). Work in semi-supervised estimation has also contributed evidence that hand-annotations are useful for training alignment models (Fraser and Marcu, 2006; Fraser and Marcu, 2007). The ITG grammar formalism, the corresponding word alignment class, and inference procedures for the class have also been explored extensively (Wu, 1997; Zhang and Gildea, 2005; Cherry and Lin, 2007; Zhang et al., 2008). At the intersection of these lines of work, discriminative ITG models have also been proposed, including one-to-one alignment models (Cherry and Lin, 2006) and block models (Haghighi et al., 2009). Our model directly extends this research agenda with first-class possible links, overlapping phrasal rule features, and an extraction-level loss function.

Kääriäinen (2009) trains a *translation* model discriminatively using features on overlapping phrase pairs. That work differs from ours in that it uses fixed word alignments and focuses on translation model estimation, while we focus on alignment and translate using standard relative frequency estimators.

Deng and Zhou (2009) present an alignment combination technique that uses phrasal features. Our approach differs in two ways. First, their approach is tightly coupled to the input alignments, while we perform a full search over the space of ITG alignments. Also, their approach uses greedy search, while our search is optimal aside from pruning and beaming. Despite these differences, their strong results reinforce our claim that phrase-level information is useful for alignment.

6 Experiments

We evaluate our extraction set model by the bispans it predicts, the word alignments it generates, and the translations generated by two end-to-end systems. Table 1 compares the five systems described below, including three baselines. All supervised aligners were optimized for bispan F_5.

Unsupervised Baseline: GIZA++. We trained GIZA++ (Och and Ney, 2003) using the default parameters included with the Moses training script (Koehn et al., 2007). The designated regimen concludes by Viterbi aligning under Model 4 in both directions. We combined these alignments with

the *grow-diag* heuristic (Koehn et al., 2003).

Unsupervised Baseline: Joint HMM. We trained and combined two HMM alignment models (Ney and Vogel, 1996) using the Berkeley Aligner.[7] We initialized the HMM model parameters with jointly trained Model 1 parameters (Liang et al., 2006), combined word-to-word posteriors by averaging (soft union), and decoded with the competitive thresholding heuristic of DeNero and Klein (2007), yielding a state-of-the-art unsupervised baseline.

Supervised Baseline: Block ITG. We discriminatively trained a block ITG aligner with only sure links, using block terminal productions up to 3 words by 3 words in size. This supervised baseline is a reimplementation of the MIRA-trained model of Haghighi et al. (2009). We use the same features and parser implementation for this model as we do for our extraction set model to ensure a clean comparison. To remain within the alignment class, MIRA updates this model toward a pseudo-gold alignment with only sure links. This model does not score any overlapping bispans.

Extraction Set Coarse Pass. We add possible links to the output of the block ITG model by adding the mixed terminal block productions described in Section 2.3. This model scores overlapping phrasal rules contained within terminal blocks that result from including or excluding possible links. However, this model does not score bispans that cross bracketing of ITG derivations.

Full Extraction Set Model. Our full model includes possible links and features on extraction sets for phrasal bispans with a maximum size of 3. Model inference is performed using the coarse-to-fine scheme described in Section 4.2.

6.1 Data

In this paper, we focus exclusively on Chinese-to-English translation. We performed our discriminative training and alignment evaluations using a hand-aligned portion of the NIST MT02 test set, which consists of 150 training and 191 test sentences (Ayan and Dorr, 2006). We trained the baseline HMM on 11.3 million words of FBIS newswire data, a comparable dataset to those used in previous alignment evaluations on our test set (DeNero and Klein, 2007; Haghighi et al., 2009).

[7] http://code.google.com/p/berkeleyaligner

Our end-to-end translation experiments were tuned and evaluated on sentences up to length 40 from the NIST MT04 and MT05 test sets. For these experiments, we trained on a 22.1 million word parallel corpus consisting of sentences up to length 40 of newswire data from the GALE program, subsampled from a larger data set to promote overlap with the tune and test sets. This corpus also includes a bilingual dictionary. To improve performance, we retrained our aligner on a retokenized version of the hand-annotated data to match the tokenization of our corpus.[8] We trained a language model with Kneser-Ney smoothing on 262 million words of newswire using SRILM (Stolcke, 2002).

6.2 Word and Phrase Alignment

The first panel of Table 1 gives a word-level evaluation of all five aligners. We use the alignment error rate (AER) measure: precision is the fraction of sure links in the system output that are sure or possible in the reference, and recall is the fraction of sure links in the reference that the system outputs as sure. For this evaluation, possible links produced by our extraction set models are ignored. The full extraction set model performs the best by a small margin, although it was not tuned for word alignment.

The second panel gives a phrasal rule-level evaluation, which measures the degree to which these aligners matched the extraction sets of hand-annotated alignments, $R_3(\mathcal{A}_t)$.[9] To compete fairly, all models were evaluated on the full extraction sets induced by the word alignments they predicted. Again, the extraction set model outperformed the baselines, particularly on the F_5 measure for which these systems were trained.

Our coarse pass extraction set model performed nearly as well as the full model. We believe these models perform similarly for two reasons. First, most of the information needed to predict an extraction set can be inferred from word links and phrasal rules contained within ITG terminal productions. Second, the coarse-to-fine inference may be constraining the full phrasal model to predict similar output to the coarse model. This similarity persists in translation experiments.

[8] All alignment results are reported under the annotated data set's original tokenization.

[9] While pseudo-gold approximations to the annotation were used for training, the evaluation is always performed relative to the original human annotation.

		Word			Bispan				BLEU	
		Pr	Rc	AER	Pr	Rc	F_1	F_5	Joshua	Moses
Baseline models	GIZA++	72.5	71.8	27.8	69.4	45.4	54.9	46.0	33.8	32.6
	Joint HMM	84.0	76.9	19.6	69.5	59.5	64.1	59.9	34.5	33.2
	Block ITG	83.4	83.8	16.4	**75.8**	62.3	68.4	62.8	34.7	33.6
Extraction set models	Coarse Pass	82.2	**84.2**	16.9	70.0	72.9	71.4	72.8	35.7	34.2
	Full Model	**84.7**	84.0	**15.6**	69.0	**74.2**	**71.6**	**74.0**	**35.9**	**34.4**

Table 1: Experimental results demonstrate that the full extraction set model outperforms supervised and unsupervised baselines in evaluations of word alignment quality, extraction set quality, and translation. In *word* and *bispan* evaluations, GIZA++ did not have access to a dictionary while all other methods did. In the *BLEU* evaluation, all systems used a bilingual dictionary included in the training corpus. The *BLEU* evaluation of supervised systems also included rule counts from the Joint HMM to compensate for parse failures.

6.3 Translation Experiments

We evaluate the alignments predicted by our model using two publicly available, open-source, state-of-the-art translation systems. Moses is a phrase-based system with lexicalized reordering (Koehn et al., 2007). Joshua (Li et al., 2009) is an implementation of Hiero (Chiang, 2007) using a suffix-array-based grammar extraction approach (Lopez, 2007).

Both of these systems take word alignments as input, and neither of these systems accepts possible links in the alignments they consume. To interface with our extraction set models, we produced three sets of sure-only alignments from our model predictions: one that omitted possible links, one that converted all possible links to sure links, and one that includes each possible link with 0.5 probability. These three sets were aggregated and rules were extracted from all three.

The training set we used for MT experiments is quite heterogenous and noisy compared to our alignment test sets, and the supervised aligners did not handle certain sentence pairs in our parallel corpus well. In some cases, pruning based on consistency with the HMM caused parse failures, which in turn caused training sentences to be skipped. To account for these issues, we added counts of phrasal rules extracted from the baseline HMM to the counts produced by supervised aligners.

In Moses, our extraction set model predicts the set of phrases extracted by the system, and so the estimation techniques for the alignment model and translation model both share a common underlying representation: extraction sets. Empirically, we observe a BLEU score improvement of 1.2 over the best unsupervised baseline and 0.8 over the block ITG supervised baseline (Papineni et al., 2002).

In Joshua, hierarchical rule extraction is based upon phrasal rule extraction, but abstracts away sub-phrases to create a grammar. Hence, the extraction sets we predict are closely linked to the representation that this system uses to translate. The extraction model again outperformed both unsupervised and supervised baselines, by 1.4 BLEU and 1.2 BLEU respectively.

7 Conclusion

Our extraction set model serves to coordinate the alignment and translation model components of a statistical translation system by unifying their representations. Moreover, our model provides an effective alternative to phrase alignment models that choose a particular phrase segmentation; instead, we predict many overlapping phrases, both large and small, that are mutually consistent. In future work, we look forward to developing extraction set models for richer formalisms, including hierarchical grammars.

Acknowledgments

This project is funded in part by BBN under DARPA contract HR0011-06-C-0022 and by the NSF under grant 0643742. We thank the anonymous reviewers for their helpful comments.

References

Necip Fazil Ayan and Bonnie J. Dorr. 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proceedings of*

the Annual Conference of the Association for Computational Linguistics.

Necip Fazil Ayan, Bonnie J. Dorr, and Christof Monz. 2005. Neuralign: combining word alignments using neural networks. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Alexandra Birch, Chris Callison-Burch, and Miles Osborne. 2006. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings of the Conference for the Association for Machine Translation in the Americas*.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Eugene Charniak and Sharon Caraballo. 1998. New figures of merit for best-first probabilistic chart parsing. In *Computational Linguistics*.

Colin Cherry and Dekang Lin. 2006. Soft syntactic constraints for word alignment through discriminative training. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Colin Cherry and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics Workshop on Syntax and Structure in Statistical Translation*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*.

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

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

John DeNero and Dan Klein. 2008. The complexity of phrase alignment problems. In *Proceedings of the Annual Conference of the Association for Computational Linguistics: Short Paper Track*.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings of the NAACL Workshop on Statistical Machine Translation*.

John DeNero, Alexandre Bouchard-Cote, and Dan Klein. 2008. Sampling alignment structure under a bayesian translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yonggang Deng and Bowen Zhou. 2009. Optimizing word alignment combination for phrase table training. In *Proceedings of the Annual Conference of the Association for Computational Linguistics: Short Paper Track*.

Alexander Fraser and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: Leaf. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Matti Kääriäinen. 2009. Sinuhe—statistical machine translation using a globally trained conditional exponential family translation model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dan Klein and Chris Manning. 2003. A* parsing: Fast exact Viterbi parse selection. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Philipp Koehn and Barry Haddow. 2009. Edinburghs submission to all tracks of the WMT2009 shared task with reordering and speed improvements to Moses. In *Proceedings of the Workshop on Statistical Machine Translation*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Conference of the Association for Computational Linguistics: Demonstration track*.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael I. Jordan. 2006. Word alignment via quadratic assignment. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Daniel Marcu and Daniel Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*.

Robert Moore and Chris Quirk. 2007. Faster beam-search decoding for phrasal statistical machine translation. In *Proceedings of MT Summit XI*.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hermann Ney and Stephan Vogel. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the Conference on Computational linguistics*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Andreas Stolcke. 2002. Srilm an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Hao Zhang and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Hao Zhang and Daniel Gildea. 2006. Efficient search for inversion transduction grammar. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of the Annual Conference of the Association for Computational Linguistics*.

Detecting Experiences from Weblogs

Keun Chan Park, Yoonjae Jeong and Sung Hyon Myaeng
Department of Computer Science
Korea Advanced Institute of Science and Technology
{keunchan, hybris, myaeng}@kaist.ac.kr

Abstract

Weblogs are a source of human activity knowledge comprising valuable information such as facts, opinions and personal experiences. In this paper, we propose a method for mining personal experiences from a large set of weblogs. We define experience as knowledge embedded in a collection of activities or events which an individual or group has actually undergone. Based on an observation that experience-revealing sentences have a certain linguistic style, we formulate the problem of detecting experience as a classification task using various features including tense, mood, aspect, modality, experiencer, and verb classes. We also present an activity verb lexicon construction method based on theories of lexical semantics. Our results demonstrate that the activity verb lexicon plays a pivotal role among selected features in the classification performance and shows that our proposed method outperforms the baseline significantly.

1 Introduction

In traditional philosophy, human beings are known to acquire knowledge mainly by reasoning and experience. Reasoning allows us to draw a conclusion based on evidence, but people tend to believe it firmly when they experience or observe it in the physical world. Despite the fact that direct experiences play a crucial role in making a firm decision and solving a problem, people often resort to indirect experiences by reading written materials or asking around. Among many sources people resort to, the Web has become the largest one for human experiences, especially with the proliferation of weblogs.

While Web documents contain various types of information including facts, encyclopedic knowledge, opinions, and experiences in general, personal experiences tend to be found in weblogs more often than other web documents like news articles, home pages, and scientific papers. As such, we have begun to see some research efforts in mining experience-related attributes such as time, location, topic, and experiencer, and their relations from weblogs (Inui et al., 2008; Kurashima et al., 2009).

Mined experiences can be of practical use in wide application areas. For example, a collection of experiences from the people who visited a resort area would help planning what to do and how to do things correctly without having to spend time sifting through a variety of resources or rely on commercially-oriented sources. Another example would be a public service department gleaning information about how a park is being used at a specific location and time.

Experiences can be recorded around a frame like "who did what, when, where, and why" although opinions and emotions can be also linked. Therefore attributes such as location, time, and activity and their relations must be extracted by devising a method for selecting experience-containing sentences based on verbs that have a particular linguistics case frame or belong to a "do" class (Kurashima et al., 2009). However, this kind of method may extract the following sentences as containing an experience:

[1] If Jason arrives on time, I'll buy him a drink.
[2] Probably, she will laugh and dance in his funeral.
[3] Can anyone explain what is going on here?
[4] Don't play soccer on the roads!

None of the sentences contain actual experiences because hypotheses, questions, and orders have not actually happened in the real world. For experience mining, it is important to ensure a sentence mentions an event or passes a factuality test to contain experience (Inui et al., 2008).

In this paper, we focus on the problem of detecting experiences from weblogs. We formulate

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1464–1472,
Uppsala, Sweden, 11-16 July 2010.

Class	Examples
State	like, know, believe
Activity	run, swim, walk
Achievement	recognize, realize
Accomplishment	paint (a picture), build (a house)

Table 1. Vendler class examples

the problem as a classification task using various linguistic features including tense, mood, aspect, modality, experiencer, and verb classes.

Based on our observation that experience-revealing sentences tend to have a certain linguistic style (Jijkoun et al., 2010), we investigate on the roles of various features. The ability to detect experience-revealing sentences should be a precursor for ensuring the quality of extracting various elements of actual experiences.

Another issue addressed in this paper is automatic construction of a lexicon for verbs related to activities and events. While there have been well-known studies about classifying verbs based on aspectual features (Vendler, 1967), thematic roles and selectional restrictions (Fillmore, 1968; Somers, 1987; Kipper et al., 2008), valence alternations and intuitions (Levin, 1993) and conceptual structures (Fillmore and Baker, 2001), we found that none of the existing lexical resources such as Framenet (Baker et al., 2003) and Verbnet (Kipper et al., 2008) are sufficient for identifying experience-revealing verbs. We introduce a method for constructing an activity/event verb lexicon based on Vendler's theory and statistics obtained by utilizing a web search engine.

We define *experience* as knowledge embedded in a collection of activities or events which an individual or group has actually undergone[1]. It can be subjective as in opinions as well as objective, but our focus in this article lies in objective knowledge. The following sentences contain objective experiences:

[5] I ran with my wife 3 times a week until we moved to Washington, D.C.
[6] Jane and I hopped on a bus into the city centre.
[7] We went to a restaurant near the central park.

Whereas sentences like the following contain subjective knowledge:

[8] I like your new style. You're beautiful!
[9] The food was great, the interior too.

Subject knowledge has been studied extensively for various functions such as identification, polarity detection, and holder extraction under the names of opinion mining and sentiment analysis (Pang and Lee, 2008).

In summary, our contribution lies in three aspects: 1) conception of experience detection, which is a precursor for experience mining, and specific related tasks that can be tackled with a high performance machine learning based solution; 2) examination and identification of salient linguistic features for experience detection; 3) a novel lexicon construction method with identification of key features to be used for verb classification.

The remainder of the paper is organized as follows. Section 2 presents our lexicon construction method with experiments. Section 3 describes the experience detection method, including experimental setup, evaluation, and results. In Section 4, we discuss related work, before we close with conclusion and future work in Section 5.

2 Lexicon Construction

Since our definition of experience is based on activities and events, it is critical to determine whether a sentence contains a predicate describing an activity or an event. To this end, it is quite conceivable that a lexicon containing activity / event verbs would play a key role. Given that our ultimate goal is to extract experiences from a large amount of weblogs, we opt for increased coverage by automatically constructing a lexicon rather than high precision obtainable by manually crafted lexicon.

Based on the theory of Vendler (1967), we classify a given verb or a verb phrase into one of the two categories: *activity* and *state*. We consider all the verbs and verb phrases in WordNet (Fellbaum, 1998) which is the largest electronic lexical database. In addition to the linguistic schemata features based on Vendler's theory, we used thematic role features and an external knowledge feature.

2.1 Background

Vendler (1967) proposes that verb meanings can be categorized into four basic classes, *states, activities, achievements*, and *accomplishments,* depending on interactions between the verbs and their aspectual and temporal modifiers. Table 1 shows some examples for the classes.

Vendler (1967) and Dowty (1979) introduce linguistic schemata that serve as evidence for the classes.

[1] http://en.wikipedia.org/wiki/Experience_(disambiguation)

Linguistic Schemata	*bs*	*prs*	*prp*	*pts*	*ptp*
No schema	■	■	■	■	■
Progressive			■		
Force	■				
Persuade	■				
Stop			■		
For	■	■	■	■	■
Carefully	■	■	■	■	■

Table 2. Query matrix. The "■" indicates that the query is applied. *No Schema* indicates that no schema is applied when the word itself is a query. *bs, prs, prp, pts, ptp* correspond to base form, present simple (3rd person singular), present participle, past simple and past participle, respectfully.

Below are the six schemata we chose because they can be tested automatically: *progressive*, *force*, *persuade*, *stop*, *for,* and *carefully* (An asterisk denotes that the statement is awkward).

- *States* cannot occur in *progressive* tense:
  John is running.
  John is liking.*
- *States* cannot occur as complements of *force* and *persuade*:
  John forced harry to run.
  John forced harry to know.*
  John persuaded harry to know.*
- *Achievements* cannot occur as complements of *stop*:
  John stopped running.
  John stopped realizing.*
- *Achievements* cannot occur with time adverbial *for*:
  John ran for an hour.
  John realized for an hour.*
- *State* and *achievement* cannot occur with adverb *carefully*:
  John runs carefully.
  John knows carefully.*

The schemata are not perfect because verbs can shift classes due to various contextual factors such as arguments and senses. However, a verb certainly has its fundamental class that is its most *natural* category at least in its dominant use.

The four classes can further be grouped into two genuses: a genus of processes going on in time and the other that refers to non-processes. *Activity* and *accomplishment* belong to the former whereas *state* and *achievement* belong to the latter. As can be seen in table 1, *states* are rather immanent operations and *achievements* are those occur in a single moment or operations related to perception level. On the other hand, *activity* and *accomplishment* are processes (transeunt operations) in traditional philosophy. We henceforth call the first genus *activity* and the latter *state.* Our aim is to classify verbs into the two genuses.

2.2 Features based on Linguistic Schemata

We developed a relatively simple computational testing method for the schemata. Assuming that an awkward expression like, "John is liking something" won't occur frequently, for example, we generated a co-occurrence based test for the first linguistic schema using the Web as a corpus. By issuing a search query, *((be OR am OR is OR was OR were OR been) and ? ing)* where '?' represents the verb at hand, to a search engine, we can get an estimate about how the verb is likely to belong to *state*. A test can be generated for each of the schemata in a similar way.

For completeness, we considered all the verb forms (i.e., 3rd person singular present, present participle, simple past, past participle) available. However, some of the patterns cannot be applied to some forms. For example, other forms except the base form cannot come as a complement of *force* (e.g., force to runs.*). Therefore, we created a query matrix which represents all query patterns we have applied, in table 2.

Based on the query matrix in table 2, we issued queries for all the verbs and verb phrases from WordNet to a search engine. We used the Google news archive search for two reasons. First, since news articles are written rather formally compared to weblogs and other web pages, the statistics obtained for a test would be more reliable. Second, Google provides an advanced option to retrieve snippets containing the query word. Normally, a snippet is composed of 3~5 sentences.

The basic statistics we consider are *hit count*, *candidate sentence count* and *correct sentence count* which we use the notations $H_{ij}(w)$, $S_{ij}(w)$, and $C_{ij}(w)$, respectfully, where w is a word, i the linguistic schema and j the verb form from the query matrix in table 2. $H_{ij}(w)$ was directly gathered from the Google search engine. $S_{ij}(w)$ is the number of sentences containing the word w in the search result snippets. $C_{ij}(w)$ is the number of correct sentences matching the query pattern among the candidate sentences. For example, the *progressive* schema for a verb "build" can retrieve the following sentences.

[10] …, New-York, is building one of the largest …
[11] Is building an artifact?

"Building" in the first example is a progressive verb, but the one in second is a noun, which does not satisfy the linguistic schema. For a POS and grammatical check of a candidate sentence, we used the Stanford POS tagger (Toutanova et al., 2003) and Stanford dependency parser (Klein and Manning, 2003).

For each linguistic schema, we derived three features: *Absolute hit ratio*, *Relative hit ratio* and *Valid ratio* for which we use the notations $A_i(w)$, $R_i(w)$ and $V_i(w)$, respectfully, where w is a word and i a linguistic schema. The index j for summations represents the j-th verb form. They are computed as follows.

$$A_i(w) = \frac{\sum_j H_{ij}(w)}{H_i(*)}$$

$$R_i(w) = \frac{\sum_j H_{ij}(w)}{\sum_j H_{No\,Scheme}(w)} \quad (1)$$

$$V_i(w) = \frac{\sum_j C_{ij}(w)}{\sum_j S_{ij}(w)}$$

Absolute hit ratio is computes the extent to which the target word w occurs with the i-th schema over all occurrences of the schema. The denominator is the *hit count* of wild card "*" matching any single word with the schema pattern from Google (e.g., $H_I(*)$, the progressive test *hit count* is 3.82×10^8). *Relative hit ratio* computes the extent to which the target word w occurs with the i-th schema over all occurrences of the word. The denominator is the sum of all verb forms. *Valid ratio* means the fraction of correct sentences among candidate sentences. The weight of a linguistic schema increases as the *valid ratio* gets high. With the three different ratios, $A_i(w)$, $R_i(w)$ and $V_i(w)$, for each test, we can generate a total of 18 features.

2.3 Features based on case frames

Since the hit count via Google API sometimes returns unreliable results (e.g., when the query becomes too long in case of long verb phrases), we also consider additional features. While our initial observation indicated that the existing lexical resources would not be sufficient for our goal, it occurred to us that the linguistic theory behind them would be worth exploring as generating additional features for categorizing verbs for the two classes. Consider the following examples:

[12] John(D) believed(V) the story(O).
[13] John(A) hit(V) him(O) with a bat(I).

The subject of a state verb is dative (D) as in [12] whereas the subject for an action verb takes the agent (A) role. In addition, a verb with the instrument (I) role tends to be an action verb. From these observations, we can use the distribution of cases (thematic roles) for a verb in a corpus. Activity verbs are expected to have high frequency of agent and instrument roles than state verbs. Although a verb may have more than one case frame, it is possible to determine which thematic roles used more dominantly.

We utilize two major resources of lexical semantics, Verbnet (Kipper et al., 2008) based on the theory of Levin (1993), and Framenet (Baker et al., 2003), which is based on Fillmore (1968). Levin (1993) demonstrated that syntactic alternations can be the basis for groupings of verbs semantically and accord reasonably well with linguistic intuitions. Verbnet provides 274 verb classes with 23 thematic roles covering 3,769 verbs based on their alternation behaviors with thematic roles annotated. Framenet defines 978 semantic frames with 7,124 unique semantic roles, covering 11,583 words including verbs, nouns, adverbs, etc.

Using Verbnet alone does not suit our needs because it has a relatively small number of example sentences. Framenet contains a much larger number of examples but the vast number of semantic roles presents a problem. In order to get meaningful distributions for a manageable number of thematic roles, we used Semlink (Loper et al., 2007) that provides a mapping between Framenet and Verbnet and uses a total of 23 thematic roles of Verbnet for the annotated corpora of the two resources. By the mapping, we obtained distributions of the thematic roles for 2,868 unique verbs that exist in both of the resources. For example, the verb "construct" has high frequencies with *agent*, *material* and *product* roles.

2.4 Features based on how-to instructions

Ryu et al. (2010) presented a method for extracting action steps for how-to goals from eHow[2] a website containing a large number of how-to instructions. The authors attempted to extract actions comprising a verb and some ingredients like an object entity from the documents based on syntactic patterns and a CRF based model.

Since each extracted action has its probability, we can use the value as a feature for state / activity verb classification. However, a verb may appear in different contexts and can have multiple

[2] http://www.ehow.com

Feature	ME		SVM	
	Prec.	*Recall*	*Prec.*	*Recall*
All 43	68%	50%	83%	75%
Top 30	72%	52%	83%	75%
Top 20	83%	76%	85%	77%
Top 10	89%	88%	91%	78%

Table 3. Classification Performance

Class	Examples
Activity	act, battle, build, carry, chase, drive, hike, jump, kick, sky dive, tap dance, walk, …
State	admire, believe, know, like, love, …

Table 4. Classified Examples

probability values. To generate a single value for a verb, we combine multiple probability values using the following sigmoid function:

$$E(w) = \frac{1}{1+e^{-t}} \qquad t = \sum_{d \in D_w} P_d(w) \tag{2}$$

Evidence of a word w being an action in eHow is denoted as $E(w)$ where variable t is the sum of individual action probability values in D_w the set of documents from which the word w has been extracted as an action. The higher probability a word gets and the more frequent the word has been extracted as an action, the more evidence we get.

2.5 Classification

For training, we selected 80 seed verbs from Dowty's list (1979) which are representative verbs for each Vendler (1967) class. The selection was based on the lack of word sense ambiguity.

One of our classifiers is based on Maximum Entropy (ME) models that implement the intuition that the best model will be the one that is consistent with the set of constraints imposed by the evidence, but otherwise is as uniform as possible (Berger et al., 1996). ME models are widely used in natural language processing tasks for its flexibility to incorporate a diverse range of features. The other one is based on Support Vector Machine (Chang and Lin, 2001) which is the state-of-the-art algorithm for many classification tasks. We used RBF kernel with the default settings (Hsu et al., 2009) because it is been known to show moderate performance using multiple feature compositions.

The features we considered are a total of 42 real values: 18 from linguistic schemata, 23 thematic role distributions, and one from eHow. In order to examine which features are discriminative for the classification, we used two well known feature selection methods, Chi-square and information gain.

2.6 Results

Table 3 shows the classification performance values for different feature selection methods. The evaluation was done on the training data with 10-fold cross validation.

Note that the precision and recall are macro-averaged values across the two classes, *activity* and *state.* The most discriminative features were *absolute ratio* and *relative ratio* in conjunction with the *force, stop, progressive,* and *persuade* schemata, the role distribution of *experiencer*, and the eHow evidence.

It is noteworthy that eHow evidence and the distribution of *experiencer* got into the top 10. Other thematic roles did not perform well because of the data sparseness. Only a few roles (e.g., experience, agent, topic, location) among the 23 had frequency values other than 0 for many verbs. Data sparseness affected the linguistic schemata as well. Many of the verbs had zero hit counts for the *for* and *carefully* schemata. It is also interesting that the validity ratio $V_i(w)$ was not shown to be a good feature-generating statistic.

We finally trained our model with the top 10 features and classified all WordNet verbs and verb phrases. For actual construction of the lexicon, 11,416 verbs and verb phrases were classified into the two classes roughly equally. We randomly sampled 200 items and examined how accurately the classification was done. A total of 164 items were correctly classified, resulting in 82% accuracy. Some examples from the classification are shown in table 4.

A further analysis of the results show that most of the errors occurred with domain-specific verbs (e.g., *ablactate*, *alkalify*, and *transaminate* in chemistry) and multi-word verb phrases (e.g., *turn a nice dime; keep one's shoulder to the wheel*). Since many features are computed based on Web resources, rare verbs cannot be classified correctly when their hit rations are very low. The domain-specific words rarely appear in Framenet or e-how, either.

3 Experience Detection

As mentioned earlier, experience-revealing sentences tend to have a certain linguistic style.

Having converted the problem of experience detection for sentences to a classification task, we focus on the extent to which various linguistic features contribute to the performance of the binary classifier for sentences. We also explain the experimental setting for evaluation, including the classifier and the test corpus.

3.1 Linguistic features

In addition to the verb class feature available in the verb lexicon constructed automatically, we used tense, mood, aspect, modality, and experiencer features.

Verb class: The feature comes directly from the lexicon since a verb has been classified into a state or activity verb. The predicate part of the sentence to be classified for experience is looked up in the lexicon without sense disambiguation.

Tense: The tense of a sentence is important since an experience-revealing sentence tends to use past and present tense. Future tenses are not experiences in most cases. We use POS tagging (Toutanova et al., 2003) for tense determination, but since the Penn tagset provides no future tenses, they are determined by exploiting modal verbs such as "will" and future expressions such "going to".

Mood: It is one of distinctive forms that are used to signal the modal status of a sentence. We consider three mood categories: indicative, imperative and subjunctive. We determine the mood of a sentence by a small set of heuristic rules using the order of POS occurrences and punctuation marks.

Aspect: It defines the temporal flow of a verb in the activity or state. Two categories are used: *progressive* and *perfective*. This feature is determined by the POS of the predicate in a sentence.

Modality: In linguistics, modals are expressions broadly associated with notions of possibility. While modality can be classified at a fine level (e.g., epistemic and deontic), we simply determine whether or not a sentence includes a modal marker that is involved in the main predicate of the sentence. In other words, this binary feature is determined based on the existence of a model verb like "can", "shall", "must", and "may" or a phrase like "have to" or "need to". The dependency parser is used to ensure a modal marker is indeed associated with the main predicate.

Experiencer: A sentence can or cannot be treated as containing an experience depending on the subject or experiencer of the verb (note that this is different from the experiencer role in a case frame). Consider the following sentences:

[14] The stranger messed up the entire garden.
[15] His presence messed up the whole situation.

The first sentence is considered an experience since the subject is a person. However, the second sentence with the same verb is not, because the subject is a non-animate abstract concept. That is, a non-animate noun can hardly constitute an experience. In order to make a distinction, we use the dependency parser and a named-entity recognizer (Finkel et al., 2005) that can recognize person pronouns and person names.

3.2 Classification

To train our classifier, we first crawled weblogs from Wordpress[3], one of the most popular blog sites in use today. Worpress provides an interface to search blog posts with queries. In selecting experience-containing blog pots, we used location names such as Central Park, SOHO, Seoul and general place names such as airport, subway station, and restaurant because blog posts with some places are expected to describe experiences rather than facts or thoughts.

We crawled 6,000 blog posts. After deleting non-English and multi-media blog posts for which we could not obtain any meaningful text data, the number became 5,326. We randomly sampled 1,000 sentences[4] and asked three annotators to judge whether or not individual sentences are considered containing an experience based on our definition. For maximum accuracy, we decided to use only those sentences all the three annotators agreed, resulting in a total of 568 sentences.

While we tested several classifiers, we chose to use two different classifiers based on SVM and Logistic Regression for the final experimental results because they showed the best performance.

3.3 Results

For comparison purposes, we take the method of Kurashima et al. (2005) as our baseline because the method was used in subsequent studies (Kurashima et al., 2006; Kurashima et al., 2009) where experience attributes are extracted. We briefly describe the method and present how we implemented it.

The method first extracts all verbs and their dependent phrasal unit from candidate sentences.

[3] http://wordpress.com
[4] It was due to the limited human resources, but when we increased the number at a later stage, the performance increase was almost negligible.

Feature	Logistic Regression		SVM	
	Prec.	*Recall*	*Prec.*	*Recall*
Baseline	32.0%	55.1%	25.3%	44.4%
Lexicon	77.5%	76.0%	77.5%	76.0%
Tense	75.1%	75.1%	75.1%	75.1%
Mood	75.8%	60.3%	75.8%	60.3%
Aspect	26.7%	51.7%	26.7%	51.7%
Modality	79.8%	70.5%	79.8%	70.5%
Experiencer	54.3%	53.5%	54.3%	53.5%
All included	91.9%	91.7%	91.7%	91.4%

Table 5. Experience Detection Performance

Feature	Logistic Regression		SVM	
	Prec.	*Recall*	*Prec.*	*Recall*
Baseline	32.0%	55.1%	25.3%	44.4%
-Lexicon	84.6%	84.6%	83.1%	81.2%
-Tense	87.3%	87.1%	86.8%	86.5%
-Mood	89.5%	89.5%	89.3%	89.2%
-Aspect	90.8%	90.5%	89.0%	88.6%
-Modality	89.5%	89.5%	82.8%	82.8%
-Experiencer	91.5%	91.4%	91.1%	90.8%
All included	91.9%	91.7%	91.7%	91.4%

Table 6. Experience Detection Performance without Individual Features

The candidate goes through three filters before it is treated as experience-containing sentence. First, the candidates that do not have an objective case (Fillmore, 1968) are eliminated because their definition of experience as "action + object". This was done by identifying the object-indicating particle (case marker) in Japanese. Next, the candidates belonging to "become" and "be" statements based on Japanese verb types are filtered out. Finally, the candidate sentences including a verb that indicates a movement are eliminated because the main interest was to identify an activity in a place.

Although their definition of experience is somewhat different from ours (i.e., "action + object"), they used the method to generate candidate sentences from which various experience attributes are extracted. From this perspective, the method functioned like our experience detection. Put differently, the definition and the method by which it is determined were much cruder than the one we are using, which seems close to our general understanding.[5]

The three filtering steps were implemented as follows. We used the dependency parser for extracting objective cases using the *direct object* relation. The second step, however, could not be applied because there is no grammatical distinction among "do, be, become" statements in English. We had to alter this step by adopting the approach of Inui et al. (2008). The authors propose a lexicon of experience expression by collecting hyponyms from a hierarchically structured dictionary. We collected all hyponyms of words "do" and "act", from WordNet (Fellbaum, 1998). Lastly, we removed all the verbs that are under the hierarchy of "move" from WordNet.

We not only compared our results with the baseline in terms of precision and recall but also evaluated individual features for their importance in experience detection (classification). The evaluation was conducted with 10-fold cross validation. The results are shown in table 5.

The performance, especially precision, of the baseline is much lower than those of the others. The method devised for Japanese doesn't seem suitable for English. It seems that the linguistic styles shown in experience expressions are different from each other. In addition, the lexicon we constructed for the baseline (i.e., using the WordNet) contains more errors than our activity lexicon for activity verbs. Some hyponyms of an activity verb may not be activity verbs. (e.g., "appear" is a hyponym of "do").

There is almost no difference between the Logistic Regression and SVM classifiers for our methods although SVM was inferior for the baseline. The performance for the best case with all the features included is very promising, closed to 92% precision and recall. Among the features, the lexicon, i.e., verb classes, gave the best result when each is used alone, followed by *modality*, *tense*, and *mood*. *Aspect* was the worst but close to the baseline. This result is very encouraging for the automatic lexicon construction work because the lexicon plays a pivotal role in the overall performance.

In order to see the effect of including individual features in the feature set, precision and recall were measured after eliminating a particular feature from the full set. The results are shown in table 6. Although the absence of the lexicon feature hurt the performance most badly, still the performance was reasonably high (roughly 84 % in precision and recall for the Logistic Regression case). Similar to table 5, the aspect and experience features were the least contributors as the performance drops are almost negligible.

[5] This is based on our observation that the three annotators found their task of identifying experience sentences not difficulty, resulting in a high degree of agreements.

4 Related Work

Experience mining in its entirety is a relatively new area where various natural language processing and text mining techniques can play a significant role. While opinion mining or sentiment analysis, which can be considered an important part of experience mining, has been studied quite extensively (see Pang and Lee's excellent survey (2008)), another sub-area, factuality analysis, begins to gain some popularity (Inui et al., 2008; Saurí, 2008). Very few studies have focused explicitly on extracting various entities that constitute experiences (Kurashima et al., 2009) or detecting experience-containing parts of text although many NLP research areas such as named entity recognition and verb classification are strongly related. The previous work on experience detection relies on a handcrafted lexicon.

There have been a number of studies for verb classification (Fillmore, 1968; Vendler, 1967; Somers, 1982; Levin, 1993; Fillmore and Baker, 2001; Kipper et al., 2008) that are essential for construction of an activity verb lexicon, which in turn is important for experience detection. Most similar to our work was done by Siegel and McKeown (2000), who attempted to categorize verbs into state or event classes based on 14 tests similar to those of Vendler's. They attempted to compute co-occurrence statistics from a corpus. The *event* class, however, includes *activity*, *accomplishment*, and *achievement*. Similarly, Zacrone and Lenci (2008) attempted to categorize verbs in Italian into the four Vendler classes using the Vendler tests by using a tagged corpus. They focused on existence of arguments such as subject and object that should co-occur with the linguistic features in the tests.

The main difference between the previous work and ours lies in the goal and scope of the work. Since our work is specifically geared toward domain-independent experience detection, we attempted to maximize the coverage by using all the verbs in WordNet, as opposed to the verbs appearing in a particular domain-specific corpus (e.g., medicine domain) as done in the previous work. Another difference is that while we are not limited to a particular domain, we did not use extensive human-annotated corpus other than using the 80 seed verbs and existing lexical resources.

5 Conclusion and Future Work

We defined experience detection as an essential task for experience mining, which is restated as determining whether individual sentences contain experience or not. Viewing the task as a classification problem, we focused on identification and examination of various linguistic features such as verb class, tense, aspect, mood, modality, and experience, all of which were computed automatically. For verb classes, in particular, we devised a method for classifying all the verbs and verb phrases in WordNet into the *activity* and *state* classes. The experimental results show that verb and verb phrase classification method is reasonably accurate with 91% precision and 78% recall with manually constructed gold standard consisting of 80 verbs and 82% accuracy for a random sample of all the WordNet entries. For experience detection, the performance was very promising, closed to 92% in precision and recall when all the features were used. Among the features, the verb classes, or the lexicon we constructed, contributed the most.

In order to increase the coverage even further and reduce the errors in lexicon construction, i.e., verb classification, caused by data sparseness, we need to devise a different method, perhaps using domain specific resources.

Given that experience mining is a relatively new research area, there are many areas to explore. In addition to refinements of our work, our next step is to develop a method for representing and extracting actual experiences from experience-revealing sentences. Furthermore, considering that only 13% of the blog data we processed contain experiences, an interesting extension is to apply the methodology to extract other types of knowledge such as facts, which are not necessarily experiences.

Acknowledgments

This research was supported by the IT R&D program of MKE/KEIT under grant KI001877 [Locational/Societal Relation-Aware Social Media Service Technology], and by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) [NIPA-2010-C1090-1011-0008].

Reference

Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Mashuichi, and Kazuhiko Ohe. 2009. TEXT2TABLE: Medical Text Summarization System based on Named Entity

Recognition and Modality Identification. In *Proceedings of the Workshop on BioNLP*.

Collin F. Baker, Charles J. Fillmore, and Beau Cronin. 2003. The Structure of the Framenet Database. *International Journal of Lexicography*.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Mximum Entropy Approach to Natural Language Processing. *Computational Linguistics*.

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM : a Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

David R. Dowty. 1979. *Word meaning and Montague Grammar*. Reidel, Dordrecht.

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

Charles J. Fillmore. 1968. The Case for Case. In Bach and Harms (Ed.): *Universals in Linguistic Theory*.

Charles J. Fillmore and Collin F. Baker. 2001. Frame Semantics for Text Understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*.

Jenny R. Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of ACL*.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. 2009. A Practical Guide to Support Vector Classification. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Kentaro Inui, Shuya Abe, Kazuo Hara, Hiraku Morita, Chitose Sao, Megumi Eguchi, Asuka Sumida, Koji Murakami, and Suguru Matsuyoshi. 2008. Experience Mining: Building a Large-Scale Database of Personal Experiences and Opinions from Web Documents. In *Proceedings of the International Conference on Web Intelligence*.

Valentin Jijkoun, Maarten de Rijke, Wouter Weerkamp, Paul Ackermans and Gijs Geleijnse. 2010. Mining User Experiences from Online Forums: An Exploration. In *Proceedings of NAACL HLT Workshop on Computational Linguistics in a World of Social Media.*

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A Large-scale Classification of English Verbs. *Language Resources and Evaluation Journal*.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*.

Takeshi Kurashima, Ko Fujimura, and Hidenori Okuda. 2009. Discovering Association Rules on Experiences from Large-Scale Blog Entries. In *Proceedings of ECIR*.

Takeshi Kurashima, Taro Tezuka, and Katsumi Tanaka. 2005. Blog Map of Experiences: Extracting and Geographically Mapping Visitor Experiences from Urban Blogs. In *Proceedings of WISE*.

Takeshi Kurashima, Taro Tezuka, and Katsumi Tanaka. 2006. Mining and Visualizing Local Experiences from Blog Entries. In *Proceedings of DEXA*.

John Lafferty, Andew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*.

Beth Levin. 1993. *English verb classes and alternations: A Preliminary investigation*. University of Chicago press.

Edward Loper, Szu-ting Yi, and Martha Palmer. 2007. Combining Lexical Resources: Mapping Between PropBank and Verbnet. In *Proceedings of the International Workshop on Computational Linguistics*.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis, *Foundations and Trends in Information Retrieval*.

Jihee Ryu, Yuchul Jung, Kyung-min Kim and Sung H. Myaeng. 2010. Automatic Extraction of Human Activity Knowledge from Method-Describing Web Articles. In *Proceedings of the 1st Workshop on Automated Knowledge Base Construction.*

Roser Saurí. 2008. A Factuality Profiler for Eventualities in Text. PhD thesis, Brandeis University.

Eric V. Siegel and Kathleen R. McKeown. 2000. Learing Methods to Combine Linguistic Indicators: Improving Aspectual Classification and Revealing Linguistic Insights. In *Computational Linguistics*.

Harold L. Somers. 1987. *Valency and Case in Computational Linguistics*. Edinburgh University Press.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL*.

Zeno Vendler. 1967. *Linguistics in Philosophy*. Cornell University Press.

Alessandra Zarcone and Alessandro Lenci. 2008. Computational Models of Event Type Classification in Context. In *Proceedings of LREC*.

Experiments in Graph-based Semi-Supervised Learning Methods for Class-Instance Acquisition

Partha Pratim Talukdar*
Search Labs, Microsoft Research
Mountain View, CA 94043
partha@talukdar.net

Fernando Pereira
Google, Inc.
Mountain View, CA 94043
pereira@google.com

Abstract

Graph-based semi-supervised learning (SSL) algorithms have been successfully used to extract class-instance pairs from large unstructured and structured text collections. However, a careful comparison of different graph-based SSL algorithms on that task has been lacking. We compare three graph-based SSL algorithms for class-instance acquisition on a variety of graphs constructed from different domains. We find that the recently proposed MAD algorithm is the most effective. We also show that class-instance extraction can be significantly improved by adding semantic information in the form of instance-attribute edges derived from an independently developed knowledge base. All of our code and data will be made publicly available to encourage reproducible research in this area.

1 Introduction

Traditionally, named-entity recognition (NER) has focused on a small number of broad classes such as person, location, organization. However, those classes are too coarse to support important applications such as sense disambiguation, semantic matching, and textual inference in Web search. For those tasks, we need a much larger inventory of specific classes and accurate classification of terms into those classes. While supervised learning methods perform well for traditional NER, they are impractical for fine-grained classification because sufficient labeled data to train classifiers for all the classes is unavailable and would be very expensive to obtain.

To overcome these difficulties, seed-based information extraction methods have been developed over the years (Hearst, 1992; Riloff and Jones, 1999; Etzioni et al., 2005; Talukdar et al., 2006; Van Durme and Paşca, 2008). Starting with a few seed instances for some classes, these methods, through analysis of unstructured text, extract new instances of the same class. This line of work has evolved to incorporate ideas from graph-based semi-supervised learning in extraction from semi-structured text (Wang and Cohen, 2007), and in combining extractions from free text and from structured sources (Talukdar et al., 2008). The benefits of combining multiple sources have also been demonstrated recently (Pennacchiotti and Pantel, 2009).

We make the following contributions:

- Even though graph-based SSL algorithms have achieved early success in class-instance acquisition, there is no study comparing different graph-based SSL methods on this task. We address this gap with a series of experiments comparing three graph-based SSL algorithms (Section 2) on graphs constructed from several sources (Metaweb Technologies, 2009; Banko et al., 2007).
- We investigate whether semantic information in the form of instance-attribute edges derived from an independent knowledge base (Suchanek et al., 2007) can improve class-instance acquisition. The intuition behind this is that instances that share attributes are more likely to belong to the same class. We demonstrate that instance-attribute edges significantly improve the accuracy of class-instance extraction. In addition, useful class-attribute relationships are learned as a by-product of this process.
- In contrast to previous studies involving pro-

* Research carried out while at the University of Pennsylvania, Philadelphia, PA, USA.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1473–1481,
Uppsala, Sweden, 11-16 July 2010.

prietary datasets (Van Durme and Paşca, 2008; Talukdar et al., 2008; Pennacchiotti and Pantel, 2009), all of our experiments use publicly available datasets and we plan to release our code[1].

In Section 2, we review three graph-based SSL algorithms that are compared for the class-instance acquisition task in Section 3. In Section 3.6, we show how additional instance-attribute based semantic constraints can be used to improve class-instance acquisition performance. We summarize the results and outline future work in Section 4.

2 Graph-based SSL

We now review the three graph-based SSL algorithms for class inference over graphs that we have evaluated.

2.1 Notation

All the algorithms compute a soft assignment of labels to the nodes of a graph $G = (V, E, W)$, where V is the set of nodes with $|V| = n$, E is the set of edges, and W is an edge weight matrix. Out of the $n = n_l + n_u$ nodes in G, n_l nodes are labeled, while the remaining n_u nodes are unlabeled. If edge $(u, v) \notin E$, $W_{uv} = 0$. The (unnormalized) Laplacian, L, of G is given by $L = D - W$, where D is an $n \times n$ diagonal degree matrix with $D_{uu} = \sum_v W_{uv}$. Let S be an $n \times n$ diagonal matrix with $S_{uu} = 1$ iff node $u \in V$ is labeled. That is, S identifies the labeled nodes in the graph. C is the set of labels, with $|C| = m$ representing the total number of labels. Y is the $n \times m$ matrix storing training label information, if any. $\hat{Y}$ is an $n \times m$ matrix of soft label assignments, with $\hat{Y}_{vl}$ representing the score of label l on node v. A graph-based SSL computes $\hat{Y}$ from $\{G, SY\}$.

2.2 Label Propagation (LP-ZGL)

The label propagation method presented by Zhu et al. (2003), which we shall refer to as LP-ZGL in this paper, is one of the first graph-based SSL methods. The objective minimized by LP-ZGL is:

$$\min_{\hat{Y}} \sum_{l \in C} \hat{Y}_l^\top L \hat{Y}_l, \text{ s.t. } SY_l = S\hat{Y}_l \quad (1)$$

where $\hat{Y}_l$ of size $n \times 1$ is the l^{th} column of $\hat{Y}$. The constraint $SY = S\hat{Y}$ makes sure that the supervised labels are not changed during inference. The above objective can be rewritten as:

$$\sum_{l \in C} \hat{Y}_l^\top L \hat{Y}_l = \sum_{u,v \in V, l \in C} W_{uv} (\hat{Y}_{ul} - \hat{Y}_{vl})^2$$

From this, we observe that LP-ZGL penalizes any label assignment where two nodes connected by a highly weighted edge are assigned different labels. In other words, LP-ZGL prefers *smooth* labelings over the graph. This property is also shared by the two algorithms we shall review next. LP-ZGL has been the basis for much subsequent work in the graph-based SSL area, and is still one of the most effective graph-based SSL algorithms.

2.3 Adsorption

Adsorption (Baluja et al., 2008) is a graph-based SSL algorithm which has been used for open-domain class-instance acquisition (Talukdar et al., 2008). Adsorption is an iterative algorithm, where label estimates on node v in the $(t+1)^{\text{th}}$ iteration are updated using estimates from the t^{th} iteration:

$$\hat{Y}_v^{(t+1)} \leftarrow p_v^{inj} \times Y_v + p_v^{cont} \times B_v^{(t)} + p_v^{abnd} \times \mathbf{r} \quad (2)$$

where,

$$B_v^{(t)} = \sum_u \frac{W_{uv}}{\sum_{u'} W_{u'v}} \hat{Y}_u^{(t)}$$

In (2), $p_v^{inj}, p_v^{cont}, \text{and } p_v^{abnd}$ are three probabilities defined on each node $v \in V$ by Adsorption; and $\mathbf{r}$ is a vector used by Adsorption to express label uncertainty at a node. On each node v, the three probabilities sum to one, i.e., $p_v^{inj} + p_v^{cont} + p_v^{abnd} = 1$, and they are based on the random-walk interpretation of the Adsorption algorithm (Talukdar et al., 2008). The main idea of Adsorption is to control label propagation more tightly by limiting the amount of information that passes through a node. For instance, Adsorption can reduce the importance of a high-degree node v during the label inference process by increasing p_v^{abnd} on that node. For more details on these, please refer to Section 2 of (Talukdar and Crammer, 2009). In contrast to LP-ZGL, Adsorption allows labels on labeled (seed) nodes to change, which is desirable in case of noisy input labels.

[1] http://www.talukdar.net/datasets/class_inst/

2.4 Modified Adsorption (MAD)

Talukdar and Crammer (2009) introduced a modification of Adsorption called MAD, which shares Adsorption's desirable properties but can be expressed as an unconstrained optimization problem:

$$\min_{\hat{Y}} \quad \sum_{l \in C} \Big[\mu_1 \left(Y_l - \hat{Y}_l \right)^\top S \left(Y_l - \hat{Y}_l \right) + \mu_2 \hat{Y}_l^\top L' \hat{Y}_l + \mu_3 \left\| \hat{Y}_l - R_l \right\|^2 \Big] \quad (3)$$

where μ_1, μ_2, and μ_3 are hyperparameters; L' is the Laplacian of an undirected graph derived from G, but with revised edge weights; and R is an $n \times m$ matrix of per-node label prior, if any, with R_l representing the l^{th} column of R. As in Adsorption, MAD allows labels on seed nodes to change. In case of MAD, the three random-walk probabilities, p_v^{inj}, p_v^{cont}, and p_v^{abnd}, defined by Adsorption on each node are folded inside the matrices S, L', and R, respectively. The optimization problem in (3) can be solved with an efficient iterative algorithm described in detail by Talukdar and Crammer (2009).

These three algorithms are all easily parallelizable in a MapReduce framework (Talukdar et al., 2008; Rao and Yarowsky, 2009), which makes them suitable for SSL on large datasets. Additionally, all three algorithms have similar space and time complexity.

3 Experiments

We now compare the experimental performance of the three graph-based SSL algorithms reviewed in the previous section, using graphs constructed from a variety of sources described below. Following previous work (Talukdar et al., 2008), we use Mean Reciprocal Rank (MRR) as the evaluation metric in all experiments:

$$\text{MRR} = \frac{1}{|Q|} \sum_{v \in Q} \frac{1}{r_v} \quad (4)$$

where $Q \subseteq V$ is the set of test nodes, and r_v is the rank of the gold label among the labels assigned to node v. Higher MRR reflects better performance. We used iterative implementations of the graph-based SSL algorithms, and the number of iterations was treated as a hyperparameter which was tuned, along with other hyperparameters, on separate held-out sets, as detailed in a longer version of this paper. Statistics of the graphs used during experiments in this section are presented in Table 1.

3.1 Freebase-1 Graph with Pantel Classes

Table ID: people-person

Name	Place of Birth	Gender
...	...	...
Isaac Newton	Lincolnshire	Male
Bob Dylan	Duluth	Male
Johnny Cash	Kingsland	Male
...	...	...

Table ID: film-music_contributor

Name	Film_Music_Credits
...	...
Bob Dylan	No Direction Home
...	...

Figure 1: Examples of two tables from Freebase, one table is from the *people* domain while the other is from the *film* domain.

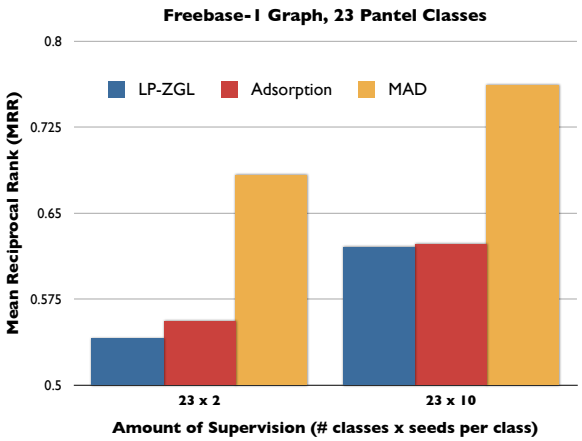


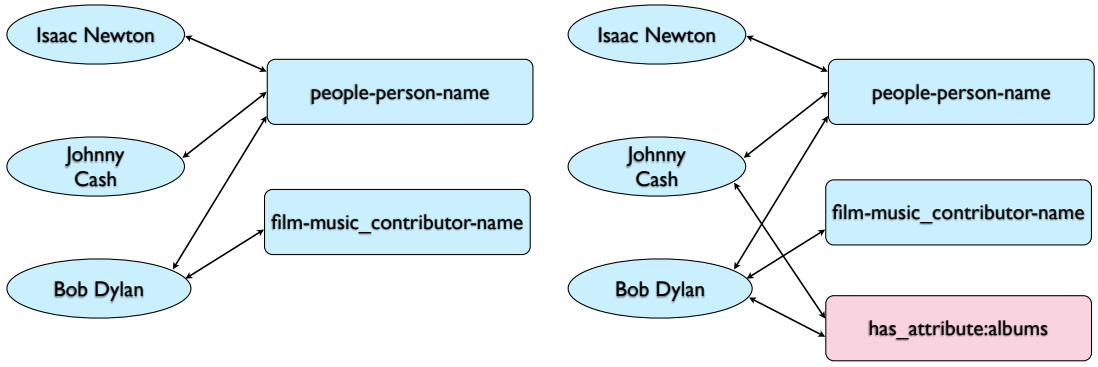
Figure 3: Comparison of three graph transduction methods on a graph constructed from the Freebase dataset (see Section 3.1), with 23 classes. All results are averaged over 4 random trials. In each group, MAD is the rightmost bar.

Freebase (Metaweb Technologies, 2009)[2] is a large collaborative knowledge base. The knowledge base harvests information from many open data sets (for instance Wikipedia and MusicBrainz), as well as from user contributions. For our current purposes, we can think of the Freebase

[2] http://www.freebase.com/

Graph	Vertices	Edges	Avg. Deg.	Min. Deg.	Max. Deg.
Freebase-1 (Section 3.1)	32970	957076	29.03	1	13222
Freebase-2 (Section 3.2)	301638	2310002	7.66	1	137553
TextRunner (Section 3.3)	175818	529557	3.01	1	2738
YAGO (Section 3.6)	142704	777906	5.45	0	74389
TextRunner + YAGO (Section 3.6)	237967	1307463	5.49	1	74389

Table 1: Statistics of various graphs used in experiments in Section 3. Some of the test instances in the YAGO graph, added for fair comparison with the TextRunner graph in Section 3.6, had no attributes in YAGO KB, and hence these instance nodes had degree 0 in the YAGO graph.



(a) (b)

Figure 2: (a) Example of a section of the graph constructed from the two tables in Figure 1. Rectangular nodes are properties, oval nodes are entities or cell values. (b) The graph in part (a) augmented with an attribute node, *has_attribue:albums*, along with the edges incident on it. This results is additional constraints for the nodes *Johnny Cash* and *Bob Dylan* to have similar labels (see Section 3.6).

dataset as a collection of relational tables, where each table is assigned a unique ID. A table consists of one or more *properties* (column names) and their corresponding *cell values* (column entries). Examples of two Freebase tables are shown in Figure 1. In this figure, *Gender* is a property in the table *people-person*, and *Male* is a corresponding cell value. We use the following process to convert the Freebase data tables into a single graph:

- Create a node for each unique cell value
- Create a node for each unique property name, where unique property name is obtained by prefixing the unique table ID to the property name. For example, in Figure 1, *people-person-gender* is a unique property name.
- Add an edge of weight 1.0 from cell-value node v to unique property node p, iff value v is present in the column corresponding to property p. Similarly, add an edge in the reverse direction.

By applying this graph construction process on the first column of the two tables in Figure 1, we end up with the graph shown in Figure 2 (a). We note that even though the resulting graph consists of edges connecting nodes of different types: cell value nodes to property nodes; the graph-based SSL methods (Section 2) can still be applied on such graphs as a cell value node and a property node connected by an edge should be assigned same or similar class labels. In other words, the label smoothness assumption (see Section 2.2) holds on such graphs.

We applied the same graph construction process on a subset of the Freebase dataset consisting of topics from 18 randomly selected domains: *astronomy, automotive, biology, book, business,*

chemistry, comic_books, computer, film, food, geography, location, people, religion, spaceflight, tennis, travel, and wine. The topics in this subset were further filtered so that only cell-value nodes with frequency 10 or more were retained. We call the resulting graph Freebase-1 (see Table 1).

Pantel et al. (2009) have made available a set of gold class-instance pairs derived from Wikipedia, which is downloadable from `http://ow.ly/13B57`. From this set, we selected all classes which had more than 10 instances overlapping with the Freebase graph constructed above. This resulted in 23 classes, which along with their overlapping instances were used as the gold standard set for the experiments in this section.

Experimental results with 2 and 10 seeds (labeled nodes) per class are shown in Figure 3. From the figure, we see that that LP-ZGL and Adsorption performed comparably on this dataset, with MAD significantly outperforming both methods.

3.2 Freebase-2 Graph with WordNet Classes

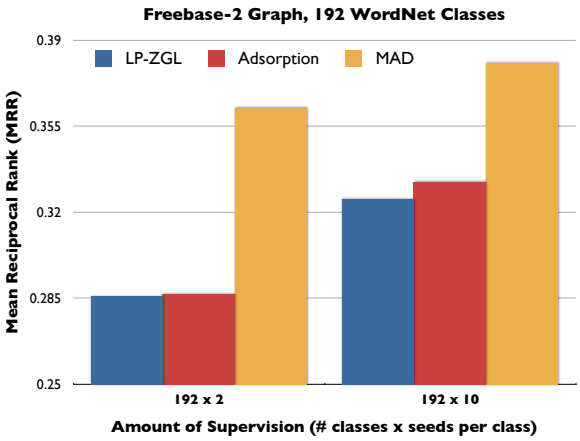


Figure 4: Comparison of graph transduction methods on a graph constructed from the Freebase dataset (see Section 3.2). All results are averaged over 10 random trials. In each group, MAD is the rightmost bar.

To evaluate how the algorithms scale up, we construct a larger graph from the same 18 domains as in Section 3.1, and using the same graph construction process. We shall call the resulting graph Freebase-2 (see Table 1). In order to scale up the number of classes, we selected all Wordnet (WN) classes, available in the YAGO KB (Suchanek et al., 2007), that had more than 100 instances overlapping with the larger Freebase graph constructed above. This resulted in 192 WN classes which we use for the experiments in this section. The reason behind imposing such frequency constraints during class selection is to make sure that each class is left with a sufficient number of instances during testing.

Experimental results comparing LP-ZGL, Adsorption, and MAD with 2 and 10 seeds per class are shown in Figure 4. A total of 292k test nodes were used for testing in the 10 seeds per class condition, showing that these methods can be applied to large datasets. Once again, we observe MAD outperforming both LP-ZGL and Adsorption. It is interesting to note that MAD with 2 seeds per class outperforms LP-ZGL and adsorption even with 10 seeds per class.

3.3 TextRunner Graph with WordNet Classes

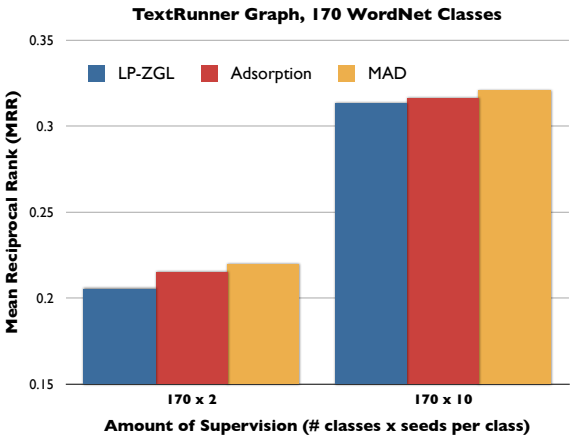


Figure 5: Comparison of graph transduction methods on a graph constructed from the hypernym tuples extracted by the TextRunner system (Banko et al., 2007) (see Section 3.3). All results are averaged over 10 random trials. In each group, MAD is the rightmost bar.

In contrast to graph construction from structured tables as in Sections 3.1, 3.2, in this section we use hypernym tuples extracted by TextRunner (Banko et al., 2007), an open domain IE system, to construct the graph. Example of a hypernym tuple extracted by TextRunner is *(http, protocol, 0.92)*, where 0.92 is the extraction confidence. To convert such a tuple into a graph, we create a node for the instance (*http*) and a node for the class (*protocol*), and then connect the nodes with two

directed edges in both directions, with the extraction confidence (*0.92*) as edge weights. The graph created with this process from TextRunner output is called the TextRunner Graph (see Table 1). As in Section 3.2, we use WordNet class-instance pairs as the gold set. In this case, we considered all WordNet classes, once again from YAGO KB (Suchanek et al., 2007), which had more than 50 instances overlapping with the constructed graph. This resulted in 170 WordNet classes being used for the experiments in this section.

Experimental results with 2 and 10 seeds per class are shown in Figure 5. The three methods are comparable in this setting, with MAD achieving the highest overall MRR.

3.4 Discussion

If we correlate the graph statistics in Table 1 with the results of sections 3.1, 3.2, and 3.3, we see that MAD is most effective for graphs with high average degree, that is, graphs where nodes tend to connect to many other nodes. For instance, the Freebase-1 graph has a high average degree of 29.03, with a corresponding large advantage for MAD over the other methods. Even though this might seem mysterious at first, it becomes clearer if we look at the objectives minimized by different algorithms. We find that the objective minimized by LP-ZGL (Equation 1) is *under-regularized*, i.e., its model parameters ($\hat{Y}$) are not constrained enough, compared to MAD (Equation 3, specifically the third term), resulting in overfitting in case of highly connected graphs. In contrast, MAD is able to avoid such overfitting because of its minimization of a well regularized objective (Equation 3). Based on this, we suggest that average degree, an easily computable structural property of the graph, may be a useful indicator in choosing which graph-based SSL algorithm should be applied on a given graph.

Unlike MAD, Adsorption does not optimize any well defined objective (Talukdar and Crammer, 2009), and hence any analysis along the lines described above is not possible. The heuristic choices made in Adsorption may have lead to its sub-optimal performance compared to MAD; we leave it as a topic for future investigation.

3.5 Effect of Per-Node Class Sparsity

For all the experiments in Sections 3.1, 3.2, and 3.6, each node was allowed to have a maximum of 15 classes during inference. After each update

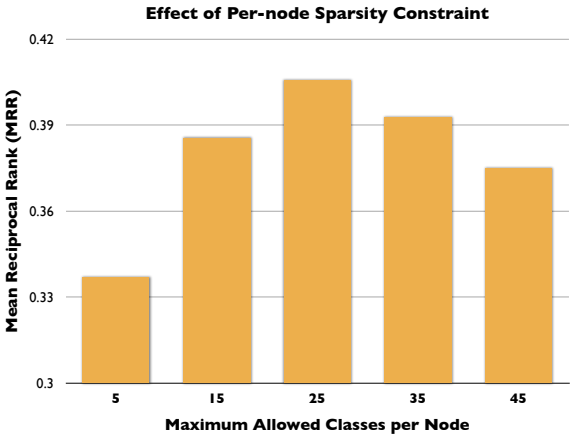


Figure 6: Effect of per node class sparsity (maximum number of classes allowed per node) during MAD inference in the experimental setting of Figure 4 (one random split).

on a node, all classes except for the top scoring 15 classes were discarded. Without such sparsity constraints, a node in a connected graph will end up acquiring all the labels injected into the graph. This is undesirable for two reasons: (1) for experiments involving a large numbers of classes (as in the previous section and in the general case of open domain IE), this increases the space requirement and also slows down inference; (2) a particular node is unlikely to belong to a large number of classes. In order to estimate the effect of such sparsity constraints, we varied the number of classes allowed per node from 5 to 45 on the graph and experimental setup of Figure 4, with 10 seeds per class. The results for MAD inference over the development split are shown in Figure 6. We observe that performance can vary significantly as the maximum number of classes allowed per node is changed, with the performance peaking at 25. This suggests that sparsity constraints during graph based SSL may have a crucial role to play, a question that needs further investigation.

3.6 TextRunner Graph with additional Semantic Constraints from YAGO

Recently, the problem of instance-attribute extraction has started to receive attention (Probst et al., 2007; Bellare et al., 2007; Pasca and Durme, 2007). An example of an instance-attribute pair is (*Bob Dylan*, *albums*). Given a set of seed instance-attribute pairs, these methods attempt to extract more instance-attribute pairs automatically

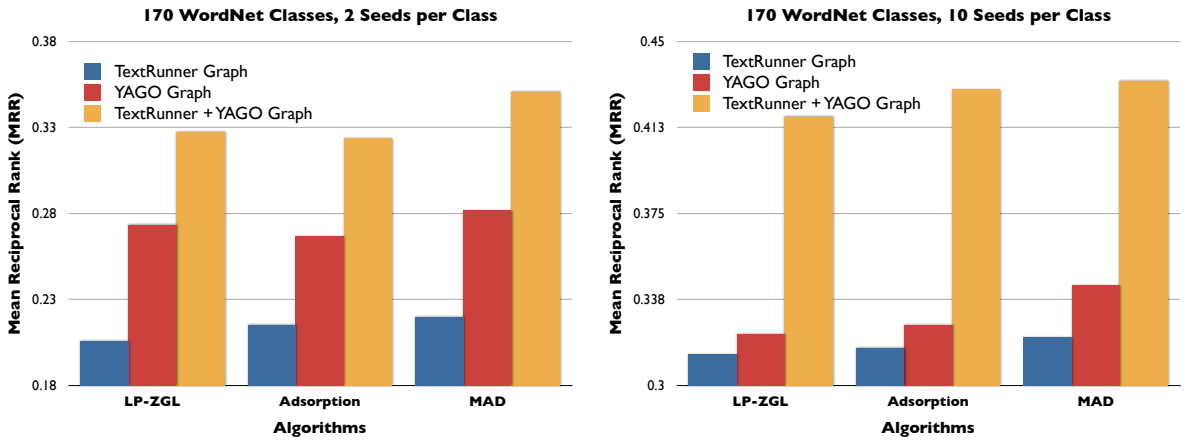


Figure 7: Comparison of class-instance acquisition performance on the three different graphs described in Section 3.6. All results are averaged over 10 random trials. Addition of YAGO attributes to the TextRunner graph significantly improves performance.

YAGO Attribute	**Top-2 WordNet Classes Assigned by MAD** (example instances for each class are shown in brackets)
has_currency	**wordnet_country_108544813 (Burma, Afghanistan)** wordnet_region_108630039 (Aosta Valley, Southern Flinders Ranges)
works_at	**wordnet_scientist_110560637 (Aage Niels Bohr, Adi Shamir)** wordnet_person_100007846 (Catherine Cornelius, Jamie White)
has_capital	**wordnet_state_108654360 (Agusan del Norte, Bali)** wordnet_region_108630039 (Aosta Valley, Southern Flinders Ranges)
born_in	**wordnet_boxer_109870208 (George Chuvalo, Fernando Montiel)** wordnet_chancellor_109906986 (Godon Brown, Bill Bryson)
has_isbn	**wordnet_book_106410904 (Past Imperfect, Berlin Diary)** wordnet_magazine_106595351 (Railway Age, Investors Chronicle)

Table 2: Top 2 (out of 170) WordNet classes assigned by MAD on 5 randomly chosen YAGO attribute nodes (out of 80) in the TextRunner + YAGO graph used in Figure 7 (see Section 3.6), with 10 seeds per class used. A few example instances of each WordNet class is shown within brackets. Top ranked class for each attribute is shown in bold.

from various sources. In this section, we explore whether class-instance assignment can be improved by incorporating new semantic constraints derived from (instance, attribute) pairs. In particular, we experiment with the following type of constraint: two instances with a common attribute are likely to belong to the same class. For example, in Figure 2 (b), instances *Johnny Cash* and *Bob Dylan* are more likely to belong to the same class as they have a common attribute, *albums*. Because of the *smooth* labeling bias of graph-based SSL methods (see Section 2.2), such constraints are naturally captured by the methods reviewed in Section 2. All that is necessary is the introduction of bidirectional (instance, attribute) edges to the graph, as shown in Figure 2 (b).

In Figure 7, we compare class-instance acquisition performance of the three graph-based SSL methods (Section 2) on the following three graphs (also see Table 1):

TextRunner Graph: Graph constructed from the hypernym tuples extracted by TextRunner, as in Figure 5 (Section 3.3), with 175k vertices and 529k edges.

YAGO Graph: Graph constructed from the (instance, attribute) pairs obtained from the YAGO KB (Suchanek et al., 2007), with 142k nodes and 777k edges.

TextRunner + YAGO Graph: Union of the

two graphs above, with 237k nodes and 1.3m edges.

In all experimental conditions with 2 and 10 seeds per class in Figure 7, we observe that the three methods consistently achieved the best performance on the TextRunner + YAGO graph. This suggests that addition of attribute based semantic constraints from YAGO to the TextRunner graph results in a better connected graph which in turn results in better inference by the graph-based SSL algorithms, compared to using either of the sources, i.e., TextRunner output or YAGO attributes, in isolation. This further illustrates the advantage of aggregating information across sources (Talukdar et al., 2008; Pennacchiotti and Pantel, 2009). However, we are the first, to the best of our knowledge, to demonstrate the effectiveness of attributes in class-instance acquisition. We note that this work is similar in spirit to the recent work by Carlson et al. (2010) which also demonstrates the benefits of additional constraints in SSL.

Because of the label propagation behavior, graph-based SSL algorithms assign classes to all nodes reachable in the graph from at least one of the labeled instance nodes. This allows us to check the classes assigned to nodes corresponding to YAGO attributes in the TextRunner + YAGO graph, as shown in Table 2. Even though the experiments were designed for class-instance acquisition, it is encouraging to see that the graph-based SSL algorithm (MAD in Table 2) is able to learn class-attribute relationships, an important by-product that has been the focus of recent studies (Reisinger and Pasca, 2009). For example, the algorithm is able to learn that *works_at* is an attribute of the WordNet class *wordnet_scientist_110560637*, and thereby its instances (e.g. *Aage Niels Bohr, Adi Shamir*).

4 Conclusion

We have started a systematic experimental comparison of graph-based SSL algorithms for class-instance acquisition on a variety of graphs constructed from different domains. We found that MAD, a recently proposed graph-based SSL algorithm, is consistently the most effective across the various experimental conditions. We also showed that class-instance acquisition performance can be significantly improved by incorporating additional semantic constraints in the class-instance acquisition process, which for the experiments in this paper were derived from instance-attribute pairs available in an independently developed knowledge base. All the data used in these experiments was drawn from publicly available datasets and we plan to release our code[3] to foster reproducible research in this area. Topics for future work include the incorporation of other kinds of semantic constraint for improved class-instance acquisition, further investigation into per-node sparsity constraints in graph-based SSL, and moving beyond bipartite graph constructions.

Acknowledgments

We thank William Cohen for valuable discussions, and Jennifer Gillenwater, Alex Kulesza, and Gregory Malecha for detailed comments on a draft of this paper. We are also very grateful to the authors of (Banko et al., 2007), Oren Etzioni and Stephen Soderland in particular, for providing TextRunner output. This work was supported in part by NSF IIS-0447972 and DARPA HRO1107-1-0029.

References

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. *Proceedings of WWW-2008*.

M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. *Procs. of IJCAI*.

K. Bellare, P. Talukdar, G. Kumaran, F. Pereira, M. Liberman, A. McCallum, and M. Dredze. 2007. Lightly-Supervised Attribute Extraction. *NIPS 2007 Workshop on Machine Learning for Web Search*.

A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr, and T.M. Mitchell. 2010. Coupled Semi-Supervised Learning for Information Extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*, volume 2, page 110.

O. Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web - an experimental study. *Artificial Intelligence Journal*.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Fourteenth International*

[3] http://www.talukdar.net/datasets/class_inst/

Conference on Computational Linguistics, Nantes, France.

Metaweb Technologies. 2009. Freebase data dumps. *http://download.freebase.com/datadumps/*.

P. Pantel, E. Crestan, A. Borkovsky, A.M. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. *Proceedings of EMNLP-09, Singapore*.

M. Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI-07. Ferbruary, 2007*.

M. Pennacchiotti and P. Pantel. 2009. Entity Extraction via Ensemble Semantics. *Proceedings of EMNLP-09, Singapore*.

K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. 2007. Semi-supervised learning of attribute-value pairs from product descriptions. In *IJCAI-07, Ferbruary, 2007*.

D. Rao and D. Yarowsky. 2009. Ranking and Semi-supervised Classification on Large Scale Graphs Using Map-Reduce. *TextGraphs*.

J. Reisinger and M. Pasca. 2009. Bootstrapped extraction of class attributes. In *Proceedings of the 18th international conference on World wide web*, pages 1235–1236. ACM.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida.

F.M. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, page 706. ACM.

P. P. Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *ECML-PKDD*.

P. P. Talukdar, T. Brants, F. Pereira, and M. Liberman. 2006. A context pattern induction method for named entity extraction. In *Tenth Conference on Computational Natural Language Learning*, page 141.

P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. 2008. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 581–589.

B. Van Durme and M. Paşca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. *Twenty-Third AAAI Conference on Artificial Intelligence*.

R. Wang and W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. *ICML-03, 20th International Conference on Machine Learning*.

Learning Arguments and Supertypes of Semantic Relations using Recursive Patterns

Zornitsa Kozareva and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{kozareva,hovy}@isi.edu

Abstract

A challenging problem in open information extraction and text mining is the learning of the selectional restrictions of semantic relations. We propose a minimally supervised bootstrapping algorithm that uses a single seed and a recursive lexico-syntactic pattern to learn the arguments and the supertypes of a diverse set of semantic relations from the Web. We evaluate the performance of our algorithm on multiple semantic relations expressed using "verb", "noun", and "verb prep" lexico-syntactic patterns. Human-based evaluation shows that the accuracy of the harvested information is about 90%. We also compare our results with existing knowledge base to outline the similarities and differences of the granularity and diversity of the harvested knowledge.

1 Introduction

Building and maintaining knowledge-rich resources is of great importance to information extraction, question answering, and textual entailment. Given the endless amount of data we have at our disposal, many efforts have focused on mining knowledge from structured or unstructured text, including ground facts (Etzioni et al., 2005), semantic lexicons (Thelen and Riloff, 2002), encyclopedic knowledge (Suchanek et al., 2007), and concept lists (Katz et al., 2003). Researchers have also successfully harvested relations between entities, such as is-a (Hearst, 1992; Pasca, 2004) and part-of (Girju et al., 2003). The kinds of knowledge learned are generally of two kinds: ground instance facts (*New York is-a city*, *Rome is the capital of Italy*) and general relational types (*city is-a location*, *engines are part-of cars*).

A variety of NLP tasks involving inference or entailment (Zanzotto et al., 2006), including QA (Katz and Lin, 2003) and MT (Mt et al., 1988), require a slightly different form of knowledge, derived from many more relations. This knowledge is usually used to support inference and is expressed as selectional restrictions (Wilks, 1975) (namely, the types of arguments that may fill a given relation, such as *person live-in city* and *airline fly-to location*). Selectional restrictions constrain the possible fillers of a relation, and hence the possible contexts in which the patterns expressing that relation can participate in, thereby enabling sense disambiguation of both the fillers and the expression itself.

To acquire this knowledge two common approaches are employed: clustering and patterns. While clustering has the advantage of being fully unsupervised, it may or may not produce the types and granularity desired by a user. In contrast pattern-based approaches are more precise, but they typically require a handful to dozens of seeds and lexico-syntactic patterns to initiate the learning process. In a closed domain these approaches are both very promising, but when tackling an unbounded number of relations they are unrealistic. The quality of clustering decreases as the domain becomes more continuously varied and diverse, and it has proven difficult to create collections of effective patterns and high-yield seeds manually.

In addition, the output of most harvesting systems is a flat list of lexical semantic expressions such as *"New York is-a city"* and *"virus causes flu"*. However, using this knowledge in inference requires it to be formulated appropriately and organized in a semantic repository. (Pennacchiotti and Pantel, 2006) proposed an algorithm for automatically ontologizing semantic relations into WordNet. However, despite its high precision entries, WordNet's limited coverage makes it impossible for relations whose arguments are not present in WordNet to be incorporated. One would like a procedure that dynamically organizes and extends

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1482–1491,
Uppsala, Sweden, 11-16 July 2010.

its semantic repository in order to be able to accommodate all newly-harvested information, and thereby become a global semantic repository.

Given these considerations, we address in this paper the following question: *How can the selectional restrictions of semantic relations be learned automatically from the Web with minimal effort using lexico-syntactic recursive patterns?*

The contributions of the paper are as follows:

- A novel representation of semantic relations using recursive lexico-syntactic patterns.
- An automatic procedure to learn the selectional restrictions (arguments and supertypes) of semantic relations from Web data.
- An exhaustive human-based evaluation of the harvested knowledge.
- A comparison of the results with some large existing knowledge bases.

The rest of the paper is organized as follows. In the next section, we review related work. Section 3 addresses the representation of semantic relations using recursive patterns. Section 4 describes the bootstrapping mechanism that learns the selectional restrictions of the relations. Section 5 describes data collection. Section 6 discusses the obtained results. Finally, we conclude in Section 7.

2 Related Work

A substantial body of work has been done in attempts to harvest bits of semantic information, including: semantic lexicons (Riloff and Shepherd, 1997), concept lists (Lin and Pantel, 2002), is-a relations (Hearst, 1992; Etzioni et al., 2005; Pasca, 2004; Kozareva et al., 2008), part-of relations (Girju et al., 2003), and others. Knowledge has been harvested with varying success both from structured text such as Wikipedia's infoboxes (Suchanek et al., 2007) or unstructured text such as the Web (Pennacchiotti and Pantel, 2006; Yates et al., 2007). A variety of techniques have been employed, including clustering (Lin and Pantel, 2002), co-occurrence statistics (Roark and Charniak, 1998), syntactic dependencies (Pantel and Ravichandran, 2004), and lexico-syntactic patterns (Riloff and Jones, 1999; Fleischman and Hovy, 2002; Thelen and Riloff, 2002).

When research focuses on a particular relation, careful attention is paid to the pattern(s) that express it in various ways (as in most of the work above, notably (Riloff and Jones, 1999)). But it has proven a difficult task to manually find effectively different variations and alternative patterns for each relation. In contrast, when research focuses on *any* relation, as in TextRunner (Yates et al., 2007), there is no standardized manner for re-using the pattern learned. TextRunner scans sentences to obtain relation-independent lexico-syntactic patterns to extract triples of the form (*John*, *fly to*, *Prague*). The middle string denotes some (unspecified) semantic relation while the first and third denote the learned arguments of this relation. But TextRunner does not seek specific semantic relations, and does not re-use the patterns it harvests with different arguments in order to extend their yields.

Clearly, it is important to be able to specify both the actual semantic relation sought *and* use its textual expression(s) in a controlled manner for maximal benefit.

The objective of our research is to combine the strengths of the two approaches, and, in addition, to provide even richer information by automatically mapping each harvested argument to its supertype(s) (i.e., its semantic concepts). For instance, given the relation *destination* and the pattern *X flies to Y*, automatically determining that *John, Prague*) and (*John, conference*) are two valid filler instance pairs, that (*RyanAir, Prague*) is another, as well as that *person* and *airline* are supertypes of the first argument and *city* and *event* of the second. This information provides the selectional restrictions of the given semantic relation, indicating that living things like people can fly to cities and events, while non-living things like airlines fly mainly to cities. This is a significant improvement over systems that output a flat list of lexical semantic knowledge (Thelen and Riloff, 2002; Yates et al., 2007; Suchanek et al., 2007).

Knowing the sectional restrictions of a semantic relation supports inference in many applications, for example enabling more accurate information extraction. (Igo and Riloff, 2009) report that patterns like "*attack on* $\langle NP \rangle$" can learn undesirable words due to idiomatic expressions and parsing errors. Over time this becomes problematic for the bootstrapping process and leads to significant deterioration in performance. (Thelen and Riloff, 2002) address this problem by learning multiple semantic categories simultaneously, relying on the often unrealistic assumption that a word cannot belong to more than one semantic category. How-

ever, if we have at our disposal a repository of semantic relations with their selectional restrictions, the problem addressed in (Igo and Riloff, 2009) can be alleviated.

In order to obtain selectional restriction classes, (Pennacchiotti and Pantel, 2006) made an attempt to ontologize the harvested arguments of *is-a*, *part-of*, and *cause* relations. They mapped each argument of the relation into WordNet and identified the senses for which the relation holds. Unfortunately, despite its very high precision entries, WordNet is known to have limited coverage, which makes it impossible for algorithms to map the content of a relation whose arguments are not present in WordNet. To surmount this limitation, we do not use WordNet, but employ a different method of obtaining superclasses of a filler term: the inverse doubly-anchored patterns DAP^{-1} (Hovy et al., 2009), which, given two arguments, harvests its supertypes from the source corpus. (Hovy et al., 2009) show that DAP^{-1} is reliable and it enriches WordNet with additional hyponyms and hypernyms.

3 Recursive Patterns

A singly-anchored pattern contains one example of the seed term (the anchor) and one open position for the term to be learned. Most researchers use singly-anchored patterns to harvest semantic relations. Unfortunately, these patterns run out of steam very quickly. To surmount this obstacle, a handful of seeds is generally used, and helps to guarantee diversity in the extraction of new lexico-syntactic patterns (Riloff and Jones, 1999; Snow et al., 2005; Etzioni et al., 2005).

Some algorithms require ten seeds (Riloff and Jones, 1999; Igo and Riloff, 2009), while others use a variation of 5, 10, to even 25 seeds (Talukdar et al., 2008). Seeds may be chosen at random (Davidov et al., 2007; Kozareva et al., 2008), by picking the most frequent terms of the desired class (Igo and Riloff, 2009), or by asking humans (Pantel et al., 2009). As (Pantel et al., 2009) show, picking seeds that yield high numbers of different terms is difficult. Thus, when dealing with unbounded sets of relations (Banko and Etzioni, 2008), providing many seeds becomes unrealistic.

Interestingly, recent work reports a class of patterns that use only one seed to learn as much information with only one seed. (Kozareva et al., 2008; Hovy et al., 2009) introduce the so-called doubly-anchored pattern (DAP) that has two anchor seed positions "⟨*type*⟩ *such as* ⟨*seed*⟩ *and* *", plus one open position for the terms to be learned. Learned terms can then be replaced into the seed position automatically, creating a recursive procedure that is reportedly much more accurate and has much higher final yield. (Kozareva et al., 2008; Hovy et al., 2009) have successfully applied DAP for the learning of hyponyms and hypernyms of *is-a* relations and report improvements over (Etzioni et al., 2005) and (Pasca, 2004).

Surprisingly, this work was limited to the semantic relation *is-a*. No other study has described the use or effect of recursive patterns for different semantic relations. Therefore, going beyond (Kozareva et al., 2008; Hovy et al., 2009), we here introduce recursive patterns other than DAP that use only one seed to harvest the arguments and supertypes of a wide variety of relations.

(Banko and Etzioni, 2008) show that semantic relations can be expressed using a handful of relation-independent lexico-syntactic patterns. Practically, we can turn *any* of these patterns into recursive form by giving as input only one of the arguments and leaving the other one as an open slot, allowing the learned arguments to replace the initial seed argument directly. For example, for the relation "*fly to*", the following recursive patterns can be built: "* *and* ⟨*seed*⟩ *fly to* *", "⟨*seed*⟩ *and* * *fly to* *", "* *fly to* ⟨*seed*⟩ *and* *", "* *fly to* * *and* ⟨*seed*⟩", "⟨*seed*⟩ *fly to* *" or "* *fly to* ⟨*seed*⟩", where ⟨*seed*⟩ is an example like *John* or *Ryanair*, and (∗) indicates the position on which the arguments are learned. Conjunctions like *and*, *or* are useful because they express list constructions and extract arguments similar to the *seed*. Potentially, one can explore all recursive pattern variations when learning a relation and compare their yield, however this study is beyond the scope of this paper.

We are particularly interested in the usage of recursive patterns for the learning of semantic relations not only because it is a novel method, but also because recursive patterns of the DAP fashion are known to: (1) learn concepts with high precision compared to singly-anchored patterns (Kozareva et al., 2008), (2) use only one seed instance for the discovery of new previously unknown terms, and (3) harvest knowledge with minimal supervision.

4 Bootstrapping Recursive Patterns

4.1 Problem Formulation

The main goal of our research is:

> **Task Definition**: Given a seed and a semantic relation expressed using a recursive lexico-syntactic pattern, learn in bootstrapping fashion the selectional restrictions (i.e., the arguments and supertypes) of the semantic relation from an unstructured corpus such as the Web.

Figure 1 shows an example of the task and the types of information learned by our algorithm.

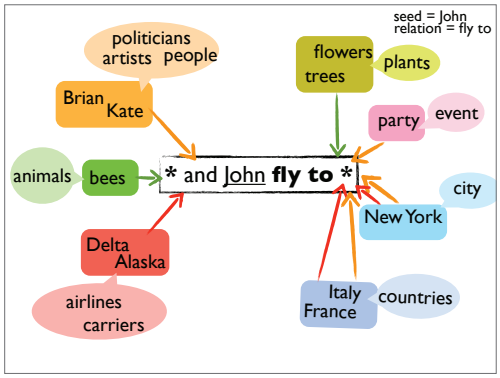


Figure 1: *Bootstrapping Recursive Patterns.*

Given a seed *John* and a semantic relation *fly to* expressed using the recursive pattern "* *and John fly to* *", our algorithm learns the left side arguments {*Brian, Kate, bees, Delta, Alaska*} and the right side arguments {*flowers, trees, party, New York, Italy, France*}. For each argument, the algorithm harvests supertypes such as {*people, artists, politicians, airlines, city, countries, plants, event*} among others. The colored links between the right and left side concepts denote the selectional restrictions of the relation. For instance, *people* fly to *events* and *countries*, but never to *trees* or *flowers*.

4.2 System Architecture

We propose a minimally supervised bootstrapping algorithm based on the framework adopted in (Kozareva et al., 2008; Hovy et al., 2009). The algorithm has two phases: *argument* harvesting and *supertype* harvesting. The final output is a ranked list of interlinked concepts which captures the selectional restrictions of the relation.

4.2.1 Argument Harvesting

In the argument extraction phase, the first bootstrapping iteration is initiated with a seed Y and a recursive pattern "X^* *and* Y *verb+prep|verb|noun* Z^*", where X^* and Z^* are the placeholders for the arguments to be learned. The pattern is submitted to Yahoo! as a web query and all unique snippets matching the query are retrieved. The newly learned and previously unexplored arguments on the X^* position are used as seeds in the subsequent iteration. The arguments on the Z^* position are stored at each iteration, but never used as seeds since the recursivity is created using the terms on X and Y. The bootstrapping process is implemented as an exhaustive breadth-first algorithm which terminates when all arguments are explored.

We noticed that despite the specific lexico-syntactic structure of the patterns, erroneous information can be acquired due to part-of-speech tagging errors or flawed facts on the Web. The challenge is to identify and separate the erroneous from the true arguments. We incorporate the harvested arguments on X and Y positions in a directed graph $G = (V, E)$, where each vertex $v \in V$ is a candidate argument and each edge $(u, v) \in E$ indicates that the argument v is generated by the argument u. An edge has weight w corresponding to the number of times the pair (u, v) is extracted from different snippets. A node u is ranked by $u = \frac{\sum_{\forall(u,v)\in E} w(u,v) + \sum_{\forall(v,u)\in E} w(v,u)}{|V|-1}$ which represents the weighted sum of the outgoing and incoming edges normalized by the total number of nodes in the graph. Intuitively, our confidence in a correct argument u increases when the argument (1) discovers and (2) is discovered by many different arguments.

Similarly, to rank the arguments standing on the Z position, we build a bipartite graph $G' = (V', E')$ that has two types of vertices. One set of vertices represents the arguments found on the Y position in the recursive pattern. We will call these V_y. The second set of vertices represents the arguments learned on the Z position. We will call these V_z. We create an edge $e'(u', v') \in E'$ between $u' \in V_y$ and $v' \in V_z$ when the argument on the Z position represented by v' was harvested by the argument on the Y position represented by u'. The weight w' of the edge indicates the number of times an argument on the Y position found Z. Vertex v' is ranked as $v' = \frac{\sum_{\forall(u',v')\in E'} w(u',v')}{|V'|-1}$. In a very large corpus, like the Web, we assume that a correct argument Z is the one that is frequently discovered by various arguments Y.

4.2.2 Supertype Harvesting

In the supertype extraction phase, we take all <X,Y> argument pairs collected during the argument harvesting stage and instantiate them in the inverse DAP^{-1} pattern *"* such as X and Y"*. The query is sent to Yahoo! as a web query and all 1000 snippets matching the pattern are retrieved. For each <X,Y> pair, the terms on the (*) position are extracted and considered as candidate supertypes.

To avoid the inclusion of erroneous supertypes, again we build a bipartite graph $G'' = (V'', E'')$. The set of vertices V_{sup} represents the supertypes, while the set of vertices V_p corresponds to the $\langle$X,Y$\rangle$ pair that produced the supertype. An edge $e''(u'', v'') \in E''$, where $u'' \in V_p$ and $v'' \in V_{sup}$ shows that the pair $\langle$X,Y$\rangle$ denoted as u'' harvested the supertype represented by v''.

For example, imagine that the argument *X**= *Ryanair* was harvested in the previous phase by the recursive pattern *"X* and EasyJet fly to Z*"*. Then the pair $\langle$*Ryanair,EasyJet*$\rangle$ forms a new Web query *"* such as Ryanair and EasyJet"* which learns the supertypes *"airlines"* and *"carriers"*. The bipartite graph has two vertices v''_1 and v''_2 for the supertypes *"airlines"* and *"carriers"*, one vertex u''_3 for the argument pair $\langle$*Ryanair, EasyJet*$\rangle$, and two edges $e''_1(u''_3, v''_1)$ and $e''_2(u''_3, v''_1)$. A vertex $v'' \in V_{sup}$ is ranked by v''=$\frac{\sum_{\forall(u'',v'')\in E''} w(u'',v'')}{|V''|-1}$. Intuitively, a supertype which is discovered multiple times by various argument pairs is ranked highly.

However, it might happen that a highly ranked supertype actually does not satisfy the selectional restrictions of the semantic relation. To avoid such situations, we further instantiate each supertype concept in the original pattern[1]. For example, *"aircompanies fly to *"* and *"carriers fly to *"*. If the candidate supertype produces many web hits for the query, then this suggests that the term is a relevant supertype.

Unfortunately, to learn the supertypes of the Z arguments, currently we have to form all possible combinations among the top 150 highly ranked concepts, because these arguments have not been learned through pairing. For each pair of Z arguments, we repeat the same procedure as described above.

[1]Except for the *"dress"* and *"person"* relations, where the targeted arguments are adjectives, and the supertypes are nouns.

5 Semantic Relations

So far, we have described the mechanism that learns from one seed and a recursive pattern the selectional restrictions of any semantic relation. Now, we are interested in evaluating the performance of our algorithm. A natural question that arises is: *"How many patterns are there?"*. (Banko and Etzioni, 2008) found that 95% of the semantic relations can be expressed using eight lexico-syntactic patterns. Space prevents us from describing all of them, therefore we focus on the three most frequent patterns which capture a large diversity of semantic relations. The relative frequency of these patterns is 37.80% for "verbs", 22.80% for "noun prep", and 16.00% for "verb prep".

5.1 Data Collection

Table 1 shows the lexico-syntactic pattern and the initial seed we used to express each semantic relation. To collect data, we ran our knowledge harvesting algorithm until complete exhaustion. For each query submitted to Yahoo!, we retrieved the top 1000 web snippets and kept only the unique ones. In total, we collected 30GB raw data which was part-of-speech tagged and used for the argument and supertype extraction. Table 1 shows the obtained results.

recursive pattern	seed	X arg	Z arg	#iter
X and Y **work for** Z	Charlie	2949	3396	20
X and Y **fly to** Z	EasyJet	772	1176	19
X and Y **go to** Z	Rita	18406	27721	13
X and Y **work in** Z	John	4142	4918	13
X and Y **work on** Z	Mary	4126	5186	7
X and Y **work at** Z	Scott	1084	1186	14
X and Y **live in** Z	Harry	8886	19698	15
X and Y **live at** Z	Donald	1102	1175	15
X and Y **live with** Z	Peter	1344	834	11
X and Y **cause** Z	virus	12790	52744	19
X and Y **celebrate**	Jim	6033	–	12
X and Y **drink**	Sam	1810	–	13
X and Y **dress**	nice	1838	–	8
X and Y **person**	scared	2984	–	17

Table 1: *Total Number of Harvested Arguments.*

An interesting characteristic of the recursive patterns is the speed of leaning which can be measured in terms of the number of unique arguments acquired during each bootstrapping iteration. Figure 2 shows the bootstrapping process for the *"cause"* and *"dress"* relations. Although both relations differ in terms of the total number of iterations and harvested items, the overall behavior of the learning curves is similar. Learning starts of very slowly and as bootstrapping progresses a

rapid growth is observed until a saturation point is reached.

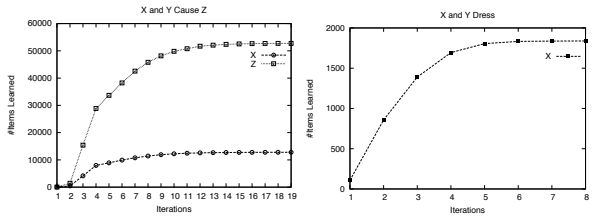


Figure 2: Items extracted in 10 iterations.

The speed of leaning is related to the connectivity behavior of the arguments of the relation. Intuitively, a densely connected graph takes shorter time (i.e., fewer iterations) to be learned, as in the *"work on"* relation, while a weakly connected network takes longer time to harvest the same amount of information, as in the *"work for"* relation.

6 Results

In this section, we evaluate the results of our knowledge harvesting algorithm. Initially, we decided to conduct an automatic evaluation comparing our results to knowledge bases that have been extracted in a similar way (i.e., through pattern application over unstructured text). However, it is not always possible to perform a complete comparison, because either researchers have not fully explored the same relations we have studied, or for those relations that overlap, the gold standard data was not available.

The online demo of TextRunner[2] (Yates et al., 2007) actually allowed us to collect the arguments for all our semantic relations. However, due to Web based query limitations, TextRunner returns only the first 1000 snippets. Since we do not have the complete and ranked output of TextRunner, comparing results in terms of recall and precision is impossible.

Turning instead to results obtained from structured sources (which one expects to have high correctness), we found that two of our relations overlap with those of the freely available ontology Yago (Suchanek et al., 2007), which was harvested from the Infoboxes tables in Wikipedia. In addition, we also had two human annotators judge as many results as we could afford, to obtain Precision. We conducted two evaluations, one for the arguments and one for the supertypes.

[2]http://www.cs.washington.edu/research/textrunner/

6.1 Human-Based Argument Evaluation

In this section, we discuss the results of the harvested arguments. For each relation, we selected the top 200 highly ranked arguments. We hired two annotators to judge their correctness. We created detailed annotation guidelines that define the labels for the arguments of the relations, as shown in Table 2. (Previously, for the same task, researchers have not conducted such an exhaustive and detailed human-based evaluation.) The annotation was conducted using the CAT system[3].

TYPE	LABEL	EXAMPLES
Correct	Person	*John, Mary*
	Role	*mother, president*
	Group	*team, Japanese*
	Physical	*yellow, shabby*
	NonPhysical	*ugly, thought*
	NonLiving	*airplane*
	Organization	*IBM, parliament*
	Location	*village, New York, in the house*
	Time	*at 5 o'clock*
	Event	*party, prom, earthquake*
	State	*sick, anrgy*
	Manner	*live in happiness*
	Medium	*work on Linux, Word*
	Fixed phrase	*go to war*
Incorrect	Error	*wrong part-of-speech tag*
	Other	*none of the above*

Table 2: *Annotation Labels.*

We allow multiple labels to be assigned to the same concept, because sometimes the concept can appear in different contexts that carry various conceptual representations. Although the labels can be easily collapsed to judge correct and incorrect terms, the fine-grained annotation shown here provides a better overview of the information learned by our algorithm.

We measured the inter-annotator agreement for all labels and relations considering that a single entry can be tagged with multiple labels. The Kappa score is around 0.80. This judgement is good enough to warrant using these human judgements to estimate the accuracy of the algorithm. We compute *Accuracy* as the number of examples tagged as $Correct$ divided by the total number of examples.

Table 4 shows the obtained results. The overall accuracy of the argument harvesting phase is 91%. The majority of the occurred errors are due to part-of-speech tagging. Table 3 shows a sample of 10 randomly selected examples from the top 200 ranked and manually annotated arguments.

[3]http://cat.ucsur.pitt.edu/default.aspx

Relation	*Arguments*
(X) Dress:	stylish, comfortable, expensive, shabby, gorgeous silver, clean, casual, Indian, black
(X) Person:	honest, caring, happy, intelligent, gifted friendly, responsible, mature, wise, outgoing
(X) Cause:	pressure, stress, fire, bacteria, cholesterol flood, ice, cocaine, injuries, wars
GoTo (Z):	school, bed, New York, the movies, the park, a bar the hospital, the church, the mall, the beach
LiveIn (Z):	peace, close proximity, harmony, Chicago, town New York, London, California, a house, Australia
WorkFor (Z):	a company, the local prison, a gangster, the show a boss, children, UNICEF, a living, Hispanics

Table 3: *Examples of Harvested Arguments.*

6.2 Comparison against Existing Resources

In this section, we compare the performance of our approach with the semantic knowledge base Yago[4] that contains 2 million entities[5], 95% of which were manually confirmed to be correct. In this study, we compare only the unique arguments of the *"live in"* and *"work at"* relations. We provide Precision scores using the following measures:

$$Pr_{Yago} = \frac{\#terms\ found\ in\ Yago}{\#terms\ harvested\ by\ system}$$

$$Pr_{Human} = \frac{\#terms\ judged\ correct\ by\ human}{\#terms\ harvested\ by\ system}$$

$$NotInYago = \#terms\ judged\ correct\ by\ human\ but\ not\ in\ Yago$$

Table 5 shows the obtained results.

We carefully analyzed those arguments that were found by one of the systems but were missing in the other. The recursive patterns learn information about non-famous entities like Peter and famous entities like Michael Jordan. In contrast, Yago contains entries mostly about famous entities, because this is the predominant knowledge in Wikipedia. For the *"live in"* relation, both repositories contain the same city and country names. However, the recursive pattern learned arguments like *pain*, *effort* which express a manner of living, and locations like *slums*, *box*. This information is missing from Yago. Similarly for the *"work at"* relation, both systems learned that people work at universities. In addition, the recursive pattern learned a diversity of company names absent from Yago.

While it is expected that our algorithm finds many terms not contained in Yago—specifically, the information not deemed worthy of inclusion in Wikipedia—we are interested in the relatively large number of terms contained in Yago but not found by our algorithm. To our knowledge, no

[4]http://www.mpi-inf.mpg.de/yago-naga/yago/

[5]Names of cities, people, organizations among others.

X WorkFor	A1	A2	WorkFor **Z**	A1	A2
Person	148	152	Organization	111	110
Role	5	7	Person	60	60
Group	12	14	Event	4	2
Organization	8	7	Time	4	5
NonPhysical	22	23	NonPhysical	18	19
Other	5	5	Other	3	4
Acc.	.98	.98	Acc.	.99	.98

X Cause	A1	A2	Cause **Z**	A1	A2
PhysicalObj	82	75	PhysicalObj	15	20
NonPhysicalObj	69	66	NonPhysicalObj	89	91
Event	21	24	Event	72	72
State	29	31	State	50	50
Other	3	4	Other	5	4
Acc.	.99	.98	Acc.	.98	.98

X GoTo	A1	A2	GoTo **Z**	A1	A2
Person	190	188	Location	163	155
Role	4	4	Event	21	30
Group	3	3	Person	11	13
NonPhysical	1	3	NonPhysical	2	1
Other	2	2	Other	3	1
Acc.	.99	.99	Acc.	.99	.99

X FlyTo	A1	A2	FlyTo **Z**	A1	A2
Person	140	139	Location	199	198
Organization	54	57	Event	1	2
NonPhysical	2	2	Person	0	0
Other	4	2	Other	0	0
Acc.	.98	.99	Acc.	1	1

X WorkOn	A1	A2	WorkOn **Z**	A1	A2
Person	173	172	Location	110	108
Role	2	3	Organization	27	25
Group	4	5	Manner	38	40
Organization	6	6	Time	4	4
NonPhysical	15	14	NonPhysical	18	21
Error	1	1	Medium	8	8
Other	1	1	Other	13	15
Acc.	.99	.99	Acc.	.94	.93

X WorkIn	A1	A2	WorkIn **Z**	A1	A2
Person	117	118	Location	104	111
Group	10	9	Organization	10	25
Organization	3	3	Manner	39	40
Fixed	3	1	Time	4	4
NonPhysical	55	59	NonPhysical	22	21
Error	12	10	Medium	8	8
Other	0	0	Error	13	15
Acc.	.94	.95	Acc.	.94	.93

X WorkAt	A1	A2	WorkAt **Z**	A1	A2
Person	193	192	Organization	189	190
Role	1	1	Manner	5	4
Group	1	1	Time	3	3
Organization	0	0	Error	3	2
Other	5	6	Other	0	1
Acc.	.98	.97	Acc.	.99	.99

X LiveIn	A1	A2	LiveIn **Z**	A1	A2
Person	185	185	Location	182	186
Role	3	4	Manner	6	8
Group	9	8	Time	1	2
NonPhysical	1	2	Fixed	5	2
Other	2	1	Other	6	2
Acc.	.99	.99	Acc.	.97	.99

X LiveAt	A1	A2	LiveAt **Z**	A1	A2
Person	196	195	Location	158	157
Role	1	1	Person	5	7
NonPhysical	0	1	Manner	1	2
Other	3	3	Error	36	34
Acc.	.99	.99	Acc.	.82	.83

X LiveWith	A1	A2	LiveWith **Z**	A1	A2
Person	188	187	Person	165	163
Role	6	6	Animal	2	4
Group	2	2	Manner	15	15
NonPhysical	2	3	NonPhysical	15	15
Other	2	2	Other	3	3
Acc.	.99	.99	Acc.	.99	.99

X Dress	A1	A2	**X** Person	A1	A2
Physical	72	59	Physical	8	2
NonPhysical	120	136	NonPhysical	188	194
Other	8	5	Other	4	4
Acc	.96	.98	Acc.	.98	.98

X Drink	A1	A2	**X** Celebrate	A1	A2
Living	165	174	Living	157	164
NonLiving	8	2	NonLiving	42	35
Error	27	24	Error	1	1
Acc	.87	.88	Acc.	.99	.99

Table 4: *Harvested Arguments.*

	Pr_{Yago}	Pr_{Human}	NotInYago
X LiveIn	.19 (2863/14705)	.58 (5165)/8886	2302
LiveIn **Z**	.10 (495/4754)	.72 (14248)/19698	13753
X WorkAt	.12(167/1399)	.88 (959)/1084	792
WorkAt **Z**	.3(15/525)	.95 (1128)/1186	1113

Table 5: *Comparison against Yago.*

other automated harvesting algorithm has ever been compared to Yago, and our results here form a baseline that we aim to improve upon. And in the future, one can build an extensive knowledge harvesting system combining the wisdom of the crowd and Wikipedia.

6.3 Human-Based Supertype Evaluation

In this section, we discuss the results of harvesting the supertypes of the learned arguments. Figure 3 shows the top 100 ranked supertypes for the *"cause"* and *"work on"* relations. The x-axis indicates a supertype, the y-axis denotes the number of different argument pairs that lead to the discovery of the supertype.

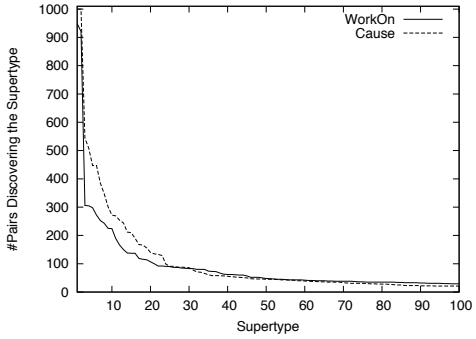


Figure 3: Ranked Supertypes.

The decline of the curve indicates that certain supertypes are preferred and shared among different argument pairs. It is interesting to note that the text on the Web prefers a small set of supertypes, and to see what they are. These most-popular harvested types tend to be the more descriptive terms. The results indicate that one does not need an elaborate supertype hierarchy to handle the selectional restrictions of semantic relations.

Since our problem definition differs from available related work, and WordNet does not contain all harvested arguments as shown in (Hovy et al., 2009), it is not possible to make a direct comparison. Instead, we conduct a manual evaluation of the most highly ranked supertypes which normally are the top 20. The overall accuracy of the supertypes for all relations is 92%. Table 6 shows the top 10 highly ranked supertypes for six of our relations.

Relation	*Arguments*
(Sup_x) **Celebrate**:	men, people, nations, angels, workers, children countries, teams, parents, teachers
(Sup_x) **Dress**:	colors, effects, color tones, activities, patterns styles, materials, size, languages, aspects
(Sup_x) **FlyTo**:	airlines, carriers, companies, giants, people competitors, political figures, stars, celebs
Cause (Sup_z):	diseases, abnormalities, disasters, processes, isses disorders, discomforts, emotions, defects, symptoms
WorkFor (Sup_z)	organizations, industries, people, markets, men automakers, countries, departments, artists, media
GoTo (Sup_z) :	countries, locations, cities, people, events men, activities, games, organizations,
FlyTo (Sup_z)	places, countries, regions, airports, destinations locations, cities, area, events

Table 6: *Examples of Harvested Supertypes.*

7 Conclusion

We propose a minimally supervised algorithm that uses only one seed example and a recursive lexico-syntactic pattern to learn in bootstrapping fashion the selectional restrictions of a large class of semantic relations. The principal contribution of the paper is to demonstrate that this kind of pattern can be applied to almost any kind of semantic relation, as long as it is expressible in a concise surface pattern, and that the recursive mechanism that allows each newly acquired term to restart harvesting automatically is a significant advance over patterns that require a handful of seeds to initiate the learning process. It also shows how one can combine free-form but undirected pattern-learning approaches like TextRunner with more-controlled but effort-intensive approaches like commonly used.

In our evaluation, we show that our algorithm is capable of extracting high quality non-trivial information from unstructured text given very restricted input (one seed). To measure the performance of our approach, we use various semantic relations expressed with three lexico-syntactic patterns. For two of the relations, we compare results with the freely available ontology Yago, and conduct a manual evaluation of the harvested terms.

We will release the annotated and the harvested data to the public to be used for comparison by other knowledge harvesting algorithms.

The success of the proposed framework opens many challenging directions. We plan to use the algorithm described in this paper to learn the selectional restrictions of numerous other relations, in order to build a rich knowledge repository

that can support a variety of applications, including textual entailment, information extraction, and question answering.

Acknowledgments

This research was supported by DARPA contract number FA8750-09-C-3705.

References

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, June.

Dmitry Davidov, Ari Rappoport, and Moshel Koppel. 2007. Fully unsupervised discovery of concept-specific relationships by web mining. In *Proc. of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 232–239, June.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th conference on Computational linguistics*, pages 539–545.

Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff. 2009. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 948–957.

Sean Igo and Ellen Riloff. 2009. Corpus-based semantic lexicon induction with web-based corroboration. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *In Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, pages 43–50.

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, and Federico Mora. 2003. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the twelfth text retrieval conference (TREC)*, pages 426–435.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056.

Dekang Lin and Patrick Pantel. 2002. Concept discovery from text. In *Proc. of the 19th international conference on Computational linguistics*, pages 1–7.

Characteristics Of Mt, John Lehrberger, Laurent Bourbeau, Philadelphia John Benjamins, and Rita Mccardell. 1988. *Machine Translation: Linguistic Characteristics of Mt Systems and General Methodology of Evaluation*. John Benjamins Publishing Co(1988-03).

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proc. of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 321–328.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947, August.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *Proc. of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145.

Marco Pennacchiotti and Patrick Pantel. 2006. Ontologizing semantic relations. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 793–800.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI '99/IAAI '99: Proceedings of the Sixteenth National Conference on Artificial intelligence.*

Ellen Riloff and Jessica Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. In *Proc. of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. In *Proceedings of the 17th international conference on Computational linguistics*, pages 1110–1116.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2008*, pages 582–590.

Michael Thelen and Ellen Riloff. 2002. A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 214–221.

Yorick Wilks. 1975. A preferential pattern-seeking, semantics for natural language inference. *Artificial Intelligence*, 6(1):53–74.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *NAACL '07: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations on XX*, pages 25–26.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Maria Teresa Pazienza. 2006. Discovering asymmetric entailment relations between verbs using selectional preferences. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 849–856.

A Transition-Based Parser for 2-Planar Dependency Structures

Carlos Gómez-Rodríguez
Departamento de Computación
Universidade da Coruña, Spain
carlos.gomez@udc.es

Joakim Nivre
Department of Linguistics and Philology
Uppsala University, Sweden
joakim.nivre@lingfil.uu.se

Abstract

Finding a class of structures that is rich enough for adequate linguistic representation yet restricted enough for efficient computational processing is an important problem for dependency parsing. In this paper, we present a transition system for 2-planar dependency trees – trees that can be decomposed into at most two planar graphs – and show that it can be used to implement a classifier-based parser that runs in linear time and outperforms a state-of-the-art transition-based parser on four data sets from the CoNLL-X shared task. In addition, we present an efficient method for determining whether an arbitrary tree is 2-planar and show that 99% or more of the trees in existing treebanks are 2-planar.

1 Introduction

Dependency-based syntactic parsing has become a widely used technique in natural language processing, and many different parsing models have been proposed in recent years (Yamada and Matsumoto, 2003; Nivre et al., 2004; McDonald et al., 2005a; Titov and Henderson, 2007; Martins et al., 2009). One of the unresolved issues in this area is the proper treatment of non-projective dependency trees, which seem to be required for an adequate representation of predicate-argument structure, but which undermine the efficiency of dependency parsing (Neuhaus and Bröker, 1997; Buch-Kromann, 2006; McDonald and Satta, 2007).

Caught between the Scylla of linguistically inadequate projective trees and the Charybdis of computationally intractable non-projective trees, some researchers have sought a middle ground by exploring classes of *mildly* non-projective dependency structures that strike a better balance between expressivity and complexity (Nivre, 2006; Kuhlmann and Nivre, 2006; Kuhlmann and Möhl, 2007; Havelka, 2007). Although these proposals seem to have a very good fit with linguistic data, in the sense that they often cover 99% or more of the structures found in existing treebanks, the development of efficient parsing algorithms for these classes has met with more limited success. For example, while both Kuhlmann and Satta (2009) and Gómez-Rodríguez et al. (2009) have shown how well-nested dependency trees with bounded gap degree can be parsed in polynomial time, the best time complexity for lexicalized parsing of this class remains a prohibitive $O(n^7)$, which makes the practical usefulness questionable.

In this paper, we explore another characterization of mildly non-projective dependency trees based on the notion of multiplanarity. This was originally proposed by Yli-Jyrä (2003) but has so far played a marginal role in the dependency parsing literature, because no algorithm was known for determining whether an arbitrary tree was m-planar, and no parsing algorithm existed for any constant value of m. The contribution of this paper is twofold. First, we present a procedure for determining the minimal number m such that a dependency tree is m-planar and use it to show that the overwhelming majority of sentences in dependency treebanks have a tree that is at most 2-planar. Secondly, we present a transition-based parsing algorithm for 2-planar dependency trees, developed in two steps. We begin by showing how the stack-based algorithm of Nivre (2003) can be generalized from projective to planar structures. We then extend the system by adding a second stack and show that the resulting system captures exactly the set of 2-planar structures. Although the contributions of this paper are mainly theoretical, we also present an empirical evaluation of the 2-planar parser, showing that it outperforms the projective parser on four data sets from the CoNLL-X shared task (Buchholz and Marsi, 2006).

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1492–1501,
Uppsala, Sweden, 11-16 July 2010.

2 Preliminaries

2.1 Dependency Graphs

Let $w = w_1 \dots w_n$ be an input string.[1] An **interval** (with endpoints i and j) of the string w is a set of the form $[i, j] = \{w_k \mid i \leq k \leq j\}$.

Definition 1. *A **dependency graph** for w is a directed graph $G = (V_w, E)$, where $V_w = [1, n]$ and $E \subseteq V_w \times V_w$.*

We call an edge (w_i, w_j) in a dependency graph G a **dependency link**[2] from w_i to w_j. We say that w_i is the **parent** (or **head**) of w_j and, conversely, that w_j is a syntactic **child** (or **dependent**) of w_i. For convenience, we write $w_i \rightarrow w_j \in E$ if the link (w_i, w_j) exists; $w_i \leftrightarrow w_j \in E$ if there is a link from w_i to w_j or from w_j to w_i; $w_i \rightarrow^* w_j \in E$ if there is a (possibly empty) directed path from w_i to w_j; and $w_i \leftrightarrow^* w_j \in E$ if there is a (possibly empty) path between w_i and w_j in the undirected graph underlying G (omitting reference to E when clear from the context). The **projection** of a node w_i, denoted $\lfloor w_i \rfloor$, is the set of reflexive-transitive dependents of w_i: $\lfloor w_i \rfloor = \{w_j \in V \mid w_i \rightarrow^* w_j\}$.

Most dependency representations do not allow arbitrary dependency graphs but typically require graphs to be acyclic and have at most one head per node. Such a graph is called a dependency *forest*.

Definition 2. *A dependency graph G for a string $w_1 \dots w_n$ is said to be a **forest** iff it satisfies:*

1. *Acyclicity: If $w_i \rightarrow^* w_j$, then not $w_j \rightarrow w_i$.*
2. *Single-head: If $w_j \rightarrow w_i$, then not $w_k \rightarrow w_i$ (for every $k \neq j$).*

Nodes in a forest that do not have a head are called **roots**. Some frameworks require that dependency forests have a unique root (i.e., are connected). Such a forest is called a **dependency tree**.

2.2 Projectivity

For reasons of computational efficiency, many dependency parsers are restricted to work with *projective* dependency structures, that is, forests in which the projection of each node corresponds to a contiguous substring of the input:

Definition 3. *A dependency forest G for a string $w_1 \dots w_n$ is **projective** iff $\lfloor w_i \rfloor$ is an interval for every word $w_i \in [1, n]$.*

Projective dependency trees correspond to the set of structures that can be induced from lexicalised context-free derivations (Kuhlmann, 2007; Gaifman, 1965). Like context-free grammars, projective dependency trees are not sufficient to represent all the linguistic phenomena observed in natural languages, but they have the advantage of being efficiently parsable: their parsing problem can be solved in cubic time with chart parsing techniques (Eisner, 1996; Gómez-Rodríguez et al., 2008), while in the case of general non-projective dependency forests, it is only tractable under strong independence assumptions (McDonald et al., 2005b; McDonald and Satta, 2007).

2.3 Planarity

The concept of *planarity* (Sleator and Temperley, 1993) is closely related to projectivity[3] and can be informally defined as the property of a dependency forest whose links can be drawn *above* the words without crossing.[4] To define planarity more formally, we first define **crossing links** as follows: let (w_i, w_k) and (w_j, w_l) be dependency links in a dependency graph G. Without loss of generality, we assume that $min(i, k) \leq min(j, l)$. Then, the links are said to be crossing if $min(i, k) < min(j, l) < max(i, k) < max(j, l)$.

Definition 4. *A dependency graph is **planar** iff it does not contain a pair of crossing links.*

2.4 Multiplanarity

The concept of planarity on its own does not seem to be very relevant as an extension of projectivity for practical dependency parsing. According to the results by Kuhlmann and Nivre (2006), most non-projective structures in dependency treebanks are also non-planar, so being able to parse planar structures will only give us a modest improvement in coverage with respect to a projective parser. However, our interest in planarity is motivated by the fact that it can be generalised to **multiplanarity** (Yli-Jyrä, 2003):

[1] For notational convenience, we will assume throughout the paper that all symbols in an input string are distinct, i.e., $i \neq j \Leftrightarrow w_i \neq w_j$. This can be guaranteed in practice by annotating each terminal symbol with its position in the input.

[2] In practice, dependency links are usually **labeled**, but to simplify the presentation we will ignore labels throughout most of the paper. However, all the results and algorithms presented can be applied to labeled dependency graphs and will be so applied in the experimental evaluation.

[3] For dependency forests that are extended with a unique artificial root located at position 0, as is commonly done, the two notions are equivalent.

[4] Planarity in the context of dependency structures is not to be confused with the homonymous concept in graph theory, which does not restrict links to be drawn above the nodes.

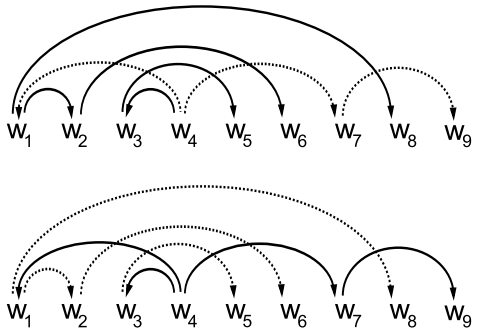


Figure 1: A 2-planar dependency structure with two different ways of distributing its links into two planes (represented by solid and dotted lines).

Definition 5. *A dependency graph $G = (V, E)$ is **m-planar** iff there exist planar dependency graphs $G_1 = (V, E_1), \ldots, G_m = (V, E_m)$ (called **planes**) such that $E = E_1 \cup \cdots \cup E_m$.*

Intuitively, we can associate planes with colours and say that a dependency graph G is m-planar if it is possible to assign one of m colours to each of its links in such a way that links with the same colour do not cross. Note that there may be multiple ways of dividing an m-planar graph into planes, as shown in the example of Figure 1.

3 Determining Multiplanarity

Several constraints on non-projective dependency structures have been proposed recently that seek a good balance between parsing efficiency and coverage of non-projective phenomena present in natural language treebanks. For example, Kuhlmann and Nivre (2006) and Havelka (2007) have shown that the vast majority of structures present in existing treebanks are well-nested and have a small gap degree (Bodirsky et al., 2005), leading to an interest in parsers for these kinds of structures (Gómez-Rodríguez et al., 2009). No similar analysis has been performed for m-planar structures, although Yli-Jyrä (2003) provides evidence that all except two structures in the Danish dependency treebank are at most 3-planar. However, his analysis is based on constraints that restrict the possible ways of assigning planes to dependency links, and he is not guaranteed to find the minimal number m for which a given structure is m-planar.

In this section, we provide a procedure for finding the minimal number m such that a dependency graph is m-planar and use it to show that the vast majority of sentences in dependency treebanks are

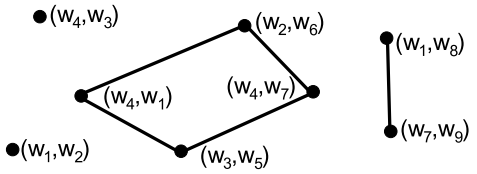


Figure 2: The crossings graph corresponding to the dependency structure of Figure 1.

at most 2-planar, with a coverage comparable to that of well-nestedness. The idea is to reduce the problem of determining whether a dependency graph $G = (V, E)$ is m-planar, for a given value of m, to a standard graph colouring problem. Consider first the following undirected graph:

$U(G) = (E, C)$ where
$C = \{\{e_i, e_j\} \mid e_i, e_j \text{ are crossing links in } G\}$

This graph, which we call the **crossings graph** of G, has one node corresponding to each link in the dependency graph G, with an undirected link between two nodes if they correspond to crossing links in G. Figure 2 shows the crossings graph of the 2-planar structure in Figure 1.

As noted in Section 2.4, a dependency graph G is m-planar if each of its links can be assigned one of m colours in such a way that links with the same colours do not cross. In terms of the crossings graph, this means that G is m-planar if each of the *nodes* of $U(G)$ can be assigned one of m colours such that no two neighbours have the same colour. This amounts to solving the well-known k-colouring problem for $U(G)$, where $k = m$.

For $k = 1$ the problem is trivial: a graph is 1-colourable only if it has no edges. For $k = 2$, the problem can be solved in time linear in the size of the graph by simple breadth-first search. Given a graph $U = (V, E)$, we pick an arbitrary node v and give it one of two colours. This forces us to give the other colour to all its neighbours, the first colour to the neighbours' neighbours, and so on. This process continues until we have processed all the nodes in the connected component of v. If this has resulted in assigning two different colours to the same node, the graph is not 2-colourable. Otherwise, we have obtained a 2-colouring of the connected component of U that contains v. If there are still unprocessed nodes, we repeat the process by arbitrarily selecting one of them, continue with the rest of the connected components, and in this way obtain a 2-colouring of the whole graph if it

Language	Structures	Non-Projective	Not Planar	Not 2-Planar	Not 3-Pl.	Not 4-pl.	Ill-nested
Arabic	2995	205 (6.84%)	158 (5.28%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	1 (0.03%)
Czech	87889	20353 (23.16%)	16660 (18.96%)	82 (0.09%)	0 (0.00%)	0 (0.00%)	96 (0.11%)
Danish	5512	853 (15.48%)	827 (15.00%)	1 (0.02%)	1 (0.02%)	0 (0.00%)	6 (0.11%)
Dutch	13349	4865 (36.44%)	4115 (30.83%)	162 (1.21%)	1 (0.01%)	0 (0.00%)	15 (0.11%)
German	39573	10927 (27.61%)	10908 (27.56%)	671 (1.70%)	0 (0.00%)	0 (0.00%)	419 (1.06%)
Portuguese	9071	1718 (18.94%)	1713 (18.88%)	8 (0.09%)	0 (0.00%)	0 (0.00%)	7 (0.08%)
Swedish	6159	293 (4.76%)	280 (4.55%)	5 (0.08%)	0 (0.00%)	0 (0.00%)	14 (0.23%)
Turkish	5510	657 (11.92%)	657 (11.92%)	10 (0.18%)	0 (0.00%)	0 (0.00%)	20 (0.36%)

Table 1: Proportion of dependency trees classified by projectivity, planarity, m-planarity and ill-nestedness in treebanks for Arabic (Hajič et al., 2004), Czech (Hajič et al., 2006), Danish (Kromann, 2003), Dutch (van der Beek et al., 2002), German (Brants et al., 2002), Portuguese (Afonso et al., 2002), Swedish (Nilsson et al., 2005) and Turkish (Oflazer et al., 2003; Atalay et al., 2003).

exists. Since this process can be completed by visiting each node and edge of the graph U once, its complexity is $O(V + E)$. The crossings graph of a dependency graph with n nodes can trivially be built in time $O(n^2)$ by checking each pair of dependency links to determine if they cross, and cannot contain more than n^2 edges, which means that we can check if the dependency graph for a sentence of length n is 2-planar in $O(n^2)$ time.

For $k > 2$, the k-colouring problem is known to be NP-complete (Karp, 1972). However, we have found this not to be a problem when measuring multiplanarity in natural language treebanks, since the effective problem size can be reduced by noting that each connected component of the crossings graph can be treated separately, and that nodes that are not part of a cycle need not be considered.[5] Given that non-projective sentences in natural language tend to have a small proportion of non-projective links (Nivre and Nilsson, 2005), the connected components of their crossings graphs are very small, and k-colourings for them can quickly be found by brute-force search.

By applying these techniques to dependency treebanks of several languages, we obtain the data shown in Table 1. As we can see, the coverage provided by the 2-planarity constraint is comparable to that of well-nestedness. In most of the treebanks, well over 99% of the sentences are 2-planar, and 3-planarity has almost total coverage. As we will see below, the class of 2-planar dependency structures not only has good coverage of linguistic phenomena in existing treebanks but is also efficiently parsable with transition-based parsing methods, making it a practically interesting subclass of non-projective dependency structures.

[5] If we have a valid colouring for all the cycles in the graph, the rest of the nodes can be safely coloured by breadth-first search as in the $k = 2$ case.

4 Parsing 1-Planar Structures

In this section, we present a deterministic linear-time parser for planar dependency structures. The parser is a variant of Nivre's arc-eager projective parser (Nivre, 2003), modified so that it can also handle graphs that are planar but not projective. As seen in Table 1, this only gives a modest improvement in coverage compared to projective parsing, so the main interest of this algorithm lies in the fact that it can be generalised to deal with 2-planar structures, as shown in the next section.

4.1 Transition Systems

In the transition-based framework of Nivre (2008), a deterministic dependency parser is defined by a non-deterministic **transition system**, specifying a set of elementary operations that can be executed during the parsing process, and an **oracle** that deterministically selects a single transition at each choice point of the parsing process.

Definition 6. *A **transition system** for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$ where*

1. *C is a set of possible parser **configurations**,*
2. *T is a set of **transitions**, each of which is a partial function $t : C \rightarrow C$,*
3. *c_s is a function that maps each input sentence w to an **initial configuration** $c_s(w) \in C$,*
4. *$C_t \subseteq C$ is a set of **terminal configurations**.*

Definition 7. *An **oracle** for a transition system $S = (C, T, c_s, C_t)$ is a function $o : C \rightarrow T$.*

An input sentence w can be parsed using a transition system $S = (C, T, c_s, C_t)$ and an oracle o by starting in the initial configuration $c_s(w)$, calling the oracle function on the current configuration c, and updating the configuration by applying the transition $o(c)$ returned by the oracle. This process is repeated until a terminal configuration is

Initial configuration: $c_s(w_1 \dots w_n) = \langle [], [w_1 \dots w_n], \emptyset \rangle$

Terminal configurations: $C_f = \{\langle \Sigma, [], A \rangle \in C\}$

Transitions:

SHIFT $\langle \Sigma, w_i|B, A\rangle \Rightarrow \langle \Sigma|w_i, B, A\rangle$

REDUCE $\langle \Sigma|w_i, B, A\rangle \Rightarrow \langle \Sigma, B, A\rangle$

LEFT-ARC $\langle \Sigma|w_i, w_j|B, A\rangle \Rightarrow \langle \Sigma|w_i, w_j|B, A \cup \{(w_j, w_i)\}\rangle$
only if $\nexists k|(w_k, w_i) \in A$ (single-head) and not $w_i \leftrightarrow^* w_j \in A$ (acyclicity).

RIGHT-ARC $\langle \Sigma|w_i, w_j|B, A\rangle \Rightarrow \langle \Sigma|w_i, w_j|B, A \cup \{(w_i, w_j)\}\rangle$
only if $\nexists k|(w_k, w_j) \in A$ (single-head) and not $w_i \leftrightarrow^* w_j \in A$ (acyclicity).

Figure 3: Transition system for planar dependency parsing.

reached, and the dependency analysis of the sentence is defined by the terminal configuration.

Each sequence of configurations that the parser can traverse from an initial configuration to a terminal configuration for some input w is called a **transition sequence**. If we associate each configuration c of a transition system $S = (C, T, c_s, C_t)$ with a dependency graph $g(c)$, we can say that S is **sound** for a class of dependency graphs $\mathcal{G}$ if, for every sentence w and transition sequence $(c_s(w), c_1, \dots, c_f)$ of S, $g(c_f)$ is in $\mathcal{G}$, and that S is **complete** for $\mathcal{G}$ if, for every sentence w and dependency graph $G \in \mathcal{G}$ for w, there is a transition sequence $(c_s(w), c_1, \dots, c_f)$ such that $g(c_f) = G$. A transition system that is sound and complete for $\mathcal{G}$ is said to be **correct** for $\mathcal{G}$.

Note that, apart from a correct transition system, a practical parser needs a good oracle to achieve the desired results, since a transition system only specifies how to reach all the possible dependency graphs that could be associated to a sentence, but not how to select the correct one. Oracles for practical parsers can be obtained by training classifiers on treebank data (Nivre et al., 2004).

4.2 A Transition System for Planar Structures

A correct transition system for the class of planar dependency forests can be obtained as a variant of the arc-eager projective system by Nivre (2003). As in that system, the set of configurations of the planar transition system is the set of all triples $c = \langle \Sigma, B, A\rangle$ such that Σ and B are disjoint lists of words from V_w (for some input w), and A is a set of dependency links over V_w. The list B, called the **buffer**, is initialised to the input string and is used to hold the words that are still to be read from the input. The list Σ, called the **stack**, is initially empty and holds words that have dependency links pending to be created. The system is shown in Figure 3, where we use the notation $\Sigma|w_i$ for a stack with top w_i and tail Σ, and we invert the notation for the buffer for clarity (i.e., $w_i|B$ is a buffer with top w_i and tail B).

The system reads the input from left to right and creates links in a left-to-right order by executing its four transitions:

1. SHIFT: pops the first (leftmost) word in the buffer, and pushes it to the stack.
2. LEFT-ARC: adds a link from the first word in the buffer to the top of the stack.
3. RIGHT-ARC: adds a link from the top of the stack to the first word in the buffer.
4. REDUCE: pops the top word from the stack, implying that we have finished building links to or from it.

Note that the planar parser's transitions are more fine-grained than those of the arc-eager projective parser by Nivre (2003), which pops the stack as part of its LEFT-ARC transition and shifts a word as part of its RIGHT-ARC transition. Forcing these actions after creating dependency links rules out structures whose root is covered by a dependency link, which are planar but not projective. In order to support these structures, we therefore simplify the ARC transitions (LEFT-ARC and RIGHT-ARC) so that they only create an arc. For the same reason, we remove the constraint in Nivre's parser by which words without a head cannot be reduced. This has the side effect of making the parser able to output cyclic graphs. Since we are interested in planar dependency *forests*, which do not contain cycles, we only apply ARC transitions after checking that there is no undirected path between the nodes to be linked. This check can be done without affecting the linear-time complexity of the

parser by storing the weakly connected component of each node in $g(c)$.

The fine-grained transitions used by this parser have also been used by Sagae and Tsujii (2008) to parse DAGs. However, the latter parser differs from ours in the constraints, since it does not allow the reduction of words without a head (disallowing forests with covered roots) and does not enforce the acyclicity constraint (which is guaranteed by post-processing the graphs to break cycles).

4.3 Correctness and Complexity

For reasons of space, we can only give a sketch of the correctness proof. We wish to prove that the planar transition system is sound and complete for the set $\mathcal{F}_p$ of all planar dependency forests. To prove soundness, we have to show that, for every sentence w and transition sequence $(c_s(w), c_1, \ldots, c_f)$, the graph $g(c_f)$ associated with c_f is in $\mathcal{F}_p$. We take the graph associated with a configuration $c = (\Sigma, B, A)$ to be $g(c) = (V_w, A)$. With this, we prove the stronger claim that $g(c) \in \mathcal{F}_p$ for every configuration c that belongs to some transition sequence starting with $c_s(w)$. This amounts to showing that in every configuration c reachable from $c_s(w)$, $g(c)$ meets the following three conditions that characterise a planar dependency forest: (1) $g(c)$ does not contain nodes with more than one head; (2) $g(c)$ is acyclic; and (3) $g(c)$ contains no crossing links. (1) is trivially guaranteed by the single-head constraint; (2) follows from (1) and the acyclicity constraint; and (3) can be established by proving that there is no transition sequence that will invoke two ARC transitions on node pairs that would create crossing links. At the point when a link from w_i to w_j is created, we know that all the words strictly located between w_i and w_j are not in the stack or in the buffer, so no links can be created to or from them.

To prove completeness, we show that every planar dependency forest $G = (V, E) \in \mathcal{F}_p$ for a sentence w can be produced by applying the oracle function that maps a configuration $\langle \Sigma | w_i, w_j | B, A \rangle$ to:

1. LEFT-ARC if $w_j \rightarrow w_i \in (E \setminus A)$,
2. RIGHT-ARC if $w_i \rightarrow w_j \in (E \setminus A)$,
3. REDUCE if $\exists x_{[x<i]}[w_x \leftrightarrow w_j \in (E \setminus A)]$,
4. SHIFT otherwise.

We show completeness by setting the following invariants on transitions traversed by the application of the oracle:

1. $\forall a, b_{[a,b<j]}[w_a \leftrightarrow w_b \in E \Rightarrow w_a \leftrightarrow w_b \in A]$
2. $[w_i \leftrightarrow w_j \in A \Rightarrow$
   $\forall k_{[i<k<j]}[w_k \leftrightarrow w_j \in E \Rightarrow w_k \leftrightarrow w_j \in A]]$
3. $\forall k_{[k<j]}[w_k \notin \Sigma \Rightarrow$
   $\forall l_{[l>k]}[w_k \leftrightarrow w_l \in E \Rightarrow w_k \leftrightarrow w_l \in A]]$

We can show that each branch of the oracle function keeps these invariants true. When we reach a terminal configuration (which always happens after a finite number of transitions, since every transition generating a configuration $c = \langle \Sigma, B, A \rangle$ decreases the value of the variant function $|E| + |\Sigma| + 2|B| - |A|$), it can be deduced from the invariant that $A = E$, which proves completeness.

The worst-case complexity of a deterministic transition-based parser is given by an upper bound on transition sequence length (Nivre, 2008). For the planar system, like its projective counterpart, the length is clearly $O(n)$ (where n is the number of input words), since there can be no more than n SHIFT transitions, n REDUCE transitions, and n ARC transitions in a transition sequence.

5 Parsing 2-Planar Structures

The planar parser introduced in the previous section can be extended to parse all 2-planar dependency structures by adding a second stack to the system and making REDUCE and ARC transitions apply to only one of the stacks at a time. This means that the set of links created in the context of each individual stack will be planar, but pairs of links created in different stacks are allowed to cross. In this way, the parser will build a 2-planar dependency forest by using each of the stacks to construct one of its two planes.

The 2-planar transition system, shown in Figure 4, has configurations of the form $\langle \Sigma_0, \Sigma_1, B, A \rangle$, where we call Σ_0 the **active stack** and Σ_1 the **inactive stack**, and the following transitions:

1. SHIFT: pops the first (leftmost) word in the buffer, and pushes it to *both* stacks.
2. LEFT-ARC: adds a link from the first word in the buffer to the top of the *active* stack.
3. RIGHT-ARC: adds a link from the top of the *active* stack to the first word in the buffer.
4. REDUCE: pops the top word from the *active* stack, implying that we have added all links to or from it on the plane tied to that stack.
5. SWITCH: makes the active stack inactive and vice versa, changing the plane the parser is working with.

Initial configuration: $c_s(w_1 \dots w_n) = \langle [], [], [w_1 \dots w_n], \emptyset \rangle$

Terminal configurations: $C_f = \{\langle \Sigma_0, \Sigma_1, [], A \rangle \in C\}$

Transitions:

SHIFT $\langle \Sigma_0, \Sigma_1, w_i|B, A\rangle \Rightarrow \langle \Sigma_0|w_i, \Sigma_1|w_i, B, A\rangle$

REDUCE $\langle \Sigma_0|w_i, \Sigma_1, B, A\rangle \Rightarrow \langle \Sigma_0, \Sigma_1, B, A\rangle$

LEFT-ARC $\langle \Sigma_0|w_i, \Sigma_1, w_j|B, A\rangle \Rightarrow \langle \Sigma_0|w_i, \Sigma_1, w_j|B, A \cup \{(w_j, w_i)\}\rangle$
only if $\nexists k \mid (w_k, w_i) \in A$ (single-head) and not $w_i \leftrightarrow^* w_j \in A$ (acyclicity).

RIGHT-ARC $\langle \Sigma_0|w_i, \Sigma_1, w_j|B, A\rangle \Rightarrow \langle \Sigma_0|w_i, \Sigma_1, w_j|B, A \cup \{(w_i, w_j)\}\rangle$
only if $\nexists k|(w_k, w_j) \in A$ (single-head) and not $w_i \leftrightarrow^* w_j \in A$ (acyclicity).

SWITCH $\langle \Sigma_0, \Sigma_1, B, A\rangle \Rightarrow \langle \Sigma_1, \Sigma_0, B, A\rangle$

Figure 4: Transition system for 2-planar dependency parsing.

5.1 Correctness and Complexity

As in the planar case, we provide a brief sketch of the proof that the transition system in Figure 4 is correct for the set $\mathcal{F}_{2p}$ of 2-planar dependency forests. Soundness follows from a reasoning analogous to the planar case, but applying the proof of planarity separately to each stack. In this way, we prove that the sets of dependency links created by linking to or from the top of each of the two stacks are always planar graphs, and thus their union (which is the dependency graph stored in A) is 2-planar. This, together with the single-head and acyclicity constraints, guarantees that the dependency graphs associated with reachable configurations are always 2-planar dependency forests.

For completeness, we assume an extended form of the transition system where transitions take the form $\langle \Sigma_0, \Sigma_1, B, A, p\rangle$, where p is a flag taking values in $\{0, 1\}$ which equals 0 for initial configurations and gets flipped by each application of a SWITCH transition. Then we show that every 2-planar dependency forest $G \in \mathcal{F}_{2p}$, with planes $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$, can be produced by this system by applying the oracle function that maps a configuration $\langle \Sigma_0|w_i, \Sigma_1, w_j|B, A, p\rangle$ to:

1. LEFT-ARC if $w_j \rightarrow w_i \in (E_p \setminus A)$,
2. RIGHT-ARC if $w_i \rightarrow w_j \in (E_p \setminus A)$,
3. REDUCE if $\exists x_{[x<i]}[w_x \leftrightarrow w_j \in (E_p \setminus A) \wedge \neg\exists y_{[x<y\leq i]}[w_y \leftrightarrow w_j \in (E_{\overline{p}} \setminus A)]]$,
4. SWITCH if $\exists x < j : (w_x, w_j)$ or $(w_j, w_x) \in (E_{\overline{p}} \setminus A)$,
5. SHIFT otherwise.

This can be shown by employing invariants analogous to the planar case, with the difference that the third invariant applies to each stack and its corresponding plane: if Σ_y is associated with the plane E_x,[6] we have:

3. $\forall k_{[k<j]}[w_k \notin \Sigma_y] \Rightarrow$
   $\forall l_{[l>k]}[w_k \leftrightarrow w_l \in E_x] \Rightarrow [w_k \leftrightarrow w_l \in A]$

Since the presence of the flag p in configurations does not affect the set of dependency graphs generated by the system, the completeness of the system extended with the flag p implies that of the system in Figure 4.

We can show that the complexity of the 2-planar system is $O(n)$ by the same kind of reasoning as for the 1-planar system, with the added complication that we must constrain the system to prevent two adjacent SWITCH transitions. In fact, without this restriction, the parser is not even guaranteed to terminate.

5.2 Implementation

In practical settings, oracles for transition-based parsers can be approximated by classifiers trained on treebank data (Nivre, 2008). To do this, we need an oracle that will generate transition sequences for gold-standard dependency graphs. In the case of the planar parser of Section 4.2, the oracle of 4.3 is suitable for this purpose. However, in the case of the 2-planar parser, the oracle used for the completeness proof in Section 5.1 cannot be used directly, since it requires the gold-standard trees to be divided into two planes in order to generate a transition sequence.

Of course, it is possible to use the algorithm presented in Section 3 to obtain a division of sentences into planes. However, for training purposes and to obtain a robust behaviour if non-2-planar

[6] The plane corresponding to each stack in a configuration changes with each SWITCH transition: Σ_x is associated with E_x in configurations where $p = 0$, and with $E_{\overline{x}}$ in those where $p = 1$.

Parser	Czech				Danish				German				Portuguese			
	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR	LAS	UAS	NPP	NPR
2-planar	79.24	85.30	68.9	**60.7**	**83.81**	88.50	**66.7**	20.0	**86.50**	**88.84**	57.1	**45.8**	87.04	**90.82**	82.8	33.8
Malt P	78.18	84.12	–	–	83.31	88.30	–	–	85.36	88.06	–	–	86.60	90.20	–	–
Malt PP	**79.80**	**85.70**	**76.7**	56.1	83.67	**88.52**	41.7	**25.0**	85.76	88.66	**58.1**	40.7	**87.08**	90.66	**83.3**	**46.2**

Table 2: Parsing accuracy for 2-planar parser in comparison to MaltParser with (PP) and without (P) pseudo-projective transformations. LAS = labeled attachment score; UAS = unlabeled attachment score; NPP = precision on non-projective arcs; NPR = recall on non-projective arcs.

sentences are found, it is more convenient that the oracle can distribute dependency links into the planes incrementally, and that it produces a distribution of links that only uses SWITCH transitions when it is strictly needed to account for non-planarity. Thus we use a more complex version of the oracle which performs a search in the crossings graph to check if a dependency link can be built on the plane of the active stack, and only performs a switch when this is not possible. This has proved to work well in practice, as will be observed in the results in the next section.

6 Empirical Evaluation

In order to get a first estimate of the empirical accuracy that can be obtained with transition-based 2-planar parsing, we have evaluated the parser on four data sets from the CoNLL-X shared task (Buchholz and Marsi, 2006): Czech, Danish, German and Portuguese. As our baseline, we take the strictly projective arc-eager transition system proposed by Nivre (2003), as implemented in the freely available MaltParser system (Nivre et al., 2006a), with and without the pseudo-projective parsing technique for recovering non-projective dependencies (Nivre and Nilsson, 2005). For the two baseline systems, we use the parameter settings used by Nivre et al. (2006b) in the original shared task, where the pseudo-projective version of MaltParser was one of the two top performing systems (Buchholz and Marsi, 2006). For our 2-planar parser, we use the same kernelized SVM classifiers as MaltParser, using the LIBSVM package (Chang and Lin, 2001), with feature models that are similar to MaltParser but extended with features defined over the second stack.[7]

In Table 2, we report labeled (LAS) and unlabeled (UAS) attachment score on the four languages for all three systems. For the two systems that are capable of recovering non-projective dependencies, we also report precision (NPP) and recall (NPR) specifically on non-projective dependency arcs. The results show that the 2-planar parser outperforms the strictly projective variant of MaltParser on all metrics for all languages, and that it performs on a par with the pseudo-projective variant with respect to both overall attachment score and precision and recall on non-projective dependencies. These results look very promising in view of the fact that very little effort has been spent on optimizing the training oracle and feature model for the 2-planar parser so far.

It is worth mentioning that the 2-planar parser has two advantages over the pseudo-projective parser. The first is simplicity, given that it is based on a single transition system and makes a single pass over the input, whereas the pseudo-projective parsing technique involves preprocessing of training data and post-processing of parser output (Nivre and Nilsson, 2005). The second is the fact that it parses a well-defined class of dependency structures, with known coverage[8], whereas no formal characterization exists of the class of structures parsable by the pseudo-projective parser.

7 Conclusion

In this paper, we have presented an efficient algorithm for deciding whether a dependency graph is 2-planar and a transition-based parsing algorithm that is provably correct for 2-planar dependency forests, neither of which existed in the literature before. In addition, we have presented empirical results showing that the class of 2-planar dependency forests includes the overwhelming majority of structures found in existing treebanks and that a deterministic classifier-based implementation of the 2-planar parser gives state-of-the-art accuracy on four different languages.

[7] Complete information about experimental settings can be found at http://stp.lingfil.uu.se/ nivre/exp/.

[8] If more coverage is desired, the 2-planar parser can be generalised to m-planar structures for larger values of m by adding additional stacks. However, this comes at the cost of more complex training models, making the practical interest of increasing m beyond 2 dubious.

Acknowledgments

The first author has been partially supported by Ministerio de Educación y Ciencia and FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, Rede Galega de Procesamento da Linguaxe e Recuperación de Información, Rede Galega de Lingüística de Corpus, Bolsas Estadías INCITE/FSE cofinanced).

References

Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1968–1703, Paris, France. ELRA.

Nart B. Atalay, Kemal Oflazer, and Bilge Say. 2003. The annotation process in the Turkish treebank. In *Proceedings of EACL Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 243–246, Morristown, NJ, USA. Association for Computational Linguistics.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Language and Computers, Computational Linguistics in the Netherlands 2001. Selected Papers from the Twelfth CLIN Meeting*, pages 8–22, Amsterdam, the Netherlands. Rodopi.

Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. In *10th Conference on Formal Grammar and 9th Meeting on Mathematics of Language*, Edinburgh, Scotland, UK.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories, September 20-21*, Sozopol, Bulgaria.

Matthias Buch-Kromann. 2006. *Discontinuous Grammar: A Model of Human Parsing and Language Acquisition*. Ph.D. thesis, Copenhagen Business School.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, San Francisco, CA, USA, August. ACL / Morgan Kaufmann.

Haim Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'08:HLT)*, pages 968–976, Morristown, NJ, USA. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 291–299.

Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. 2006. Prague Dependency Treebank 2.0. CDROM CAT: LDC2006T01, ISBN 1-58563-370-4. Linguistic Data Consortium.

Jiri Havelka. 2007. Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 608–615.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Matthias T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 217–220, Växjö, Sweden. Växjö University Press.

Marco Kuhlmann and Mathias Möhl. 2007. Mildly context-sensitive dependency languages. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 160–167.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 478–486.

Marco Kuhlmann. 2007. *Dependency Structures and Lexicalized Grammars*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 342–350.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 122–131.

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

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP 2005: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Morristown, NJ, USA. Association for Computational Linguistics.

Peter Neuhaus and Norbert Bröker. 1997. The complexity of recognition of linguistically adequate dependency grammars. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–343.

Jens Nilsson, Johan Hall, and Joakim Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proceedings of NODALIDA 2005 Special Session on Treebanks*, pages 119–132. Samfundslitteratur, Frederiksberg, Denmark, May.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *ACL '05: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 99–106, Morristown, NJ, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56, Morristown, NJ, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225.

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

Joakim Nivre. 2006. Constraints on non-projective dependency graphs. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 73–80.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics*, 34(4):513–553.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In A. Abeille (ed.), *Building and Exploiting Syntactically-annotated Corpora*, pages 261–277, Dordrecht, the Netherlands. Kluwer.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760, Morristown, NJ, USA. Association for Computational Linguistics.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. In *Proceedings of the Third International Workshop on Parsing Technologies (IWPT'93)*, pages 277–292. ACL/SIGPARSE.

Ivan Titov and James Henderson. 2007. A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pages 144–155.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Anssi Mikael Yli-Jyrä. 2003. Multiplanarity – a model for dependency structures in treebanks. In Joakim Nivre and Erhard Hinrichs, editors, *TLT 2003. Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, volume 9 of *Mathematical Modelling in Physics, Engineering and Cognitive Sciences*, pages 189–200, Växjö, Sweden, 14-15 November. Växjö University Press.

Viterbi Training for PCFGs: Hardness Results and Competitiveness of Uniform Initialization

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

Abstract

We consider the search for a maximum likelihood assignment of hidden derivations and grammar weights for a probabilistic context-free grammar, the problem approximately solved by "Viterbi training." We show that solving and even approximating Viterbi training for PCFGs is NP-hard. We motivate the use of uniform-at-random initialization for Viterbi EM as an optimal initializer in absence of further information about the correct model parameters, providing an approximate bound on the log-likelihood.

1 Introduction

Probabilistic context-free grammars are an essential ingredient in many natural language processing models (Charniak, 1997; Collins, 2003; Johnson et al., 2006; Cohen and Smith, 2009, *inter alia*). Various algorithms for training such models have been proposed, including unsupervised methods. Many of these are based on the expectation-maximization (EM) algorithm.

There are alternatives to EM, and one such alternative is Viterbi EM, also called "hard" EM or "sparse" EM (Neal and Hinton, 1998). Instead of using the parameters (which are maintained in the algorithm's current state) to find the true posterior over the derivations, Viterbi EM algorithm uses a posterior focused on the Viterbi parse of those parameters. Viterbi EM and variants have been used in various settings in natural language processing (Yejin and Cardie, 2007; Wang et al., 2007; Goldwater and Johnson, 2005; DeNero and Klein, 2008; Spitkovsky et al., 2010).

Viterbi EM can be understood as a coordinate ascent procedure that locally optimizes a function; we call this optimization goal "Viterbi training."

In this paper, we explore Viterbi training for probabilistic context-free grammars. We first show that under the assumption that P $\neq$ NP, solving and even approximating the Viterbi training problem is hard. This result holds even for hidden Markov models. We extend the main hardness result to the EM algorithm (giving an alternative proof to this known result), as well as the problem of *conditional* Viterbi training. We then describe a "competitiveness" result for uniform initialization of Viterbi EM: we show that initialization of the trees in an E-step which uses uniform distributions over the trees is optimal with respect to a certain approximate bound.

The rest of this paper is organized as follows. §2 gives background on PCFGs and introduces some notation. §3 explains Viterbi training, the declarative form of Viterbi EM. §4 describes a hardness result for Viterbi training. §5 extends this result to a hardness result of approximation and §6 further extends these results for other cases. §7 describes the advantages in using uniform-at-random initialization for Viterbi training. We relate these results to work on the k-means problem in §8.

2 Background and Notation

We assume familiarity with probabilistic context-free grammars (PCFGs). A PCFG $\boldsymbol{G}$ consists of:

- A finite set of nonterminal symbols $\mathcal{N}$;
- A finite set of terminal symbols Σ;
- For each $A \in \mathcal{N}$, a set of rewrite rules $R(A)$ of the form $A \rightarrow \alpha$, where $\alpha \in (\mathcal{N} \cup \Sigma)^*$, and $\mathcal{R} = \cup_{A \in \mathcal{N}} R(A)$;
- For each rule $A \rightarrow \alpha$, a probability $\theta_{A \rightarrow \alpha}$. The collection of probabilities is denoted $\boldsymbol{\theta}$, and they are constrained such that:

$$\begin{aligned} \forall (A \rightarrow \alpha) \in R(A), \quad \theta_{A \rightarrow \alpha} &\geq 0 \\ \forall A \in \mathcal{N}, \quad \sum_{\alpha : (A \rightarrow \alpha) \in R(A)} \theta_{A \rightarrow \alpha} &= 1 \end{aligned}$$

That is, $\boldsymbol{\theta}$ is grouped into $|\mathcal{N}|$ multinomial distributions.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1502–1511,
Uppsala, Sweden, 11-16 July 2010.

Under the PCFG, the joint probability of a string $x \in \Sigma^*$ and a grammatical derivation z is[1]

$$p(x, z \mid \boldsymbol{\theta}) = \prod_{(A\to\alpha)\in\mathcal{R}} (\theta_{A\to\alpha})^{f_{A\to\alpha}(z)} \quad (1)$$
$$= \exp \sum_{(A\to\alpha)\in\mathcal{R}} f_{A\to\alpha}(z) \log \theta_{A\to\alpha}$$

where $f_{A\to\alpha}(z)$ is a function that "counts" the number of times the rule $A \to \alpha$ appears in the derivation z. $f_A(z)$ will similarly denote the number of times that nonterminal A appears in z. Given a sample of derivations $\boldsymbol{z} = \langle z_1, \ldots, z_n \rangle$, let:

$$F_{A\to\alpha}(\boldsymbol{z}) = \sum_{i=1}^{n} f_{A\to\alpha}(z_i) \quad (2)$$

$$F_A(\boldsymbol{z}) = \sum_{i=1}^{n} f_A(z_i) \quad (3)$$

We use the following notation for $\boldsymbol{G}$:

- $L(\boldsymbol{G})$ is the set of all strings (sentences) x that can be generated using the grammar $\boldsymbol{G}$ (the "language of $\boldsymbol{G}$").
- $D(\boldsymbol{G})$ is the set of all possible derivations z that can be generated using the grammar $\boldsymbol{G}$.
- $D(\boldsymbol{G}, x)$ is the set of all possible derivations z that can be generated using the grammar $\boldsymbol{G}$ and have the yield x.

3 Viterbi Training

Viterbi EM, or "hard" EM, is an unsupervised learning algorithm, used in NLP in various settings (Yejin and Cardie, 2007; Wang et al., 2007; Goldwater and Johnson, 2005; DeNero and Klein, 2008; Spitkovsky et al., 2010). In the context of PCFGs, it aims to select parameters $\boldsymbol{\theta}$ and phrase-structure trees $\boldsymbol{z}$ jointly. It does so by iteratively updating a state consisting of $(\boldsymbol{\theta}, \boldsymbol{z})$. The state is initialized with some value, then the algorithm alternates between (i) a "hard" E-step, where the strings $x_1, \ldots, x_n$ are parsed according to a current, fixed $\boldsymbol{\theta}$, giving new values for $\boldsymbol{z}$, and (ii) an M-step, where the $\boldsymbol{\theta}$ are selected to maximize likelihood, with $\boldsymbol{z}$ fixed.

With PCFGs, the E-step requires running an algorithm such as (probabilistic) CKY or Earley's algorithm, while the M-step normalizes frequency counts $F_{A\to\alpha}(\boldsymbol{z})$ to obtain the maximum likelihood estimate's closed-form solution.

We can understand Viterbi EM as a coordinate ascent procedure that approximates the solution to the following declarative problem:

Problem 1. ViterbiTrain
Input: *$\boldsymbol{G}$ context-free grammar, $x_1, \ldots, x_n$ training instances from $L(\boldsymbol{G})$*
Output: *$\boldsymbol{\theta}$ and $z_1, \ldots, z_n$ such that*

$$(\boldsymbol{\theta}, z_1, \ldots, z_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \quad (4)$$

The optimization problem in Eq. 4 is non-convex and, as we will show in §4, hard to optimize. Therefore it is necessary to resort to approximate algorithms like Viterbi EM.

Neal and Hinton (1998) use the term "sparse EM" to refer to a version of the EM algorithm where the E-step finds the modes of hidden variables (rather than marginals as in standard EM). Viterbi EM is a variant of this, where the E-step finds the mode for each x_i's derivation, $\operatorname{argmax}_{z\in D(\boldsymbol{G},x_i)} p(x_i, z \mid \boldsymbol{\theta})$.

We will refer to

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z}) = \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \quad (5)$$

as "the objective function of ViterbiTrain."

Viterbi training and Viterbi EM are closely related to **self-training**, an important concept in semi-supervised NLP (Charniak, 1997; McClosky et al., 2006a; McClosky et al., 2006b). With self-training, the model is learned with some seed annotated data, and then iterates by labeling new, unannotated data and adding it to the original annotated training set. McClosky et al. consider self-training to be "one round of Viterbi EM" with supervised initialization using labeled seed data. We refer the reader to Abney (2007) for more details.

4 Hardness of Viterbi Training

We now describe hardness results for Problem 1. We first note that the following problem is known to be NP-hard, and in fact, NP-complete (Sipser, 2006):

Problem 2. 3-SAT
Input: *A formula $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ in conjunctive normal form, such that each clause has* 3

[1] Note that $x = \text{yield}(z)$; if the derivation is known, the string is also known. On the other hand, there may be many derivations with the same yield, perhaps even infinitely many.

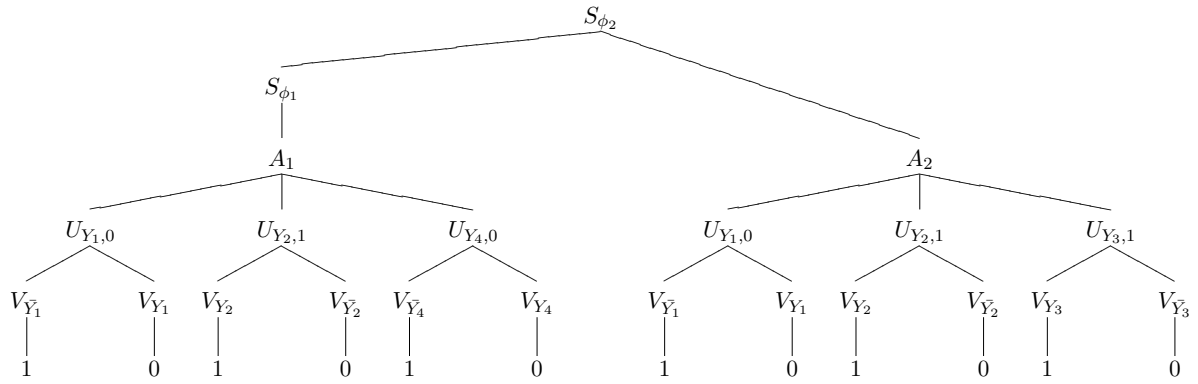


Figure 1: An example of a Viterbi parse tree which represents a satisfying assignment for $\phi = (Y_1 \vee Y_2 \vee \bar{Y_4}) \wedge (\bar{Y_1} \vee \bar{Y_2} \vee Y_3)$. In $\boldsymbol{\theta}_\phi$, all rules appearing in the parse tree have probability 1. The extracted assignment would be $Y_1 = 0, Y_2 = 1, Y_3 = 1, Y_4 = 0$. Note that there is no usage of two different rules for a single nonterminal.

literals.

Output: *1 if there is a satisfying assignment for ϕ and 0 otherwise.*

We now describe a reduction of 3-SAT to Problem 1. Given an instance of the 3-SAT problem, the reduction will, in polynomial time, create a grammar and a single string such that solving the ViterbiTrain problem for this grammar and string will yield a solution for the instance of the 3-SAT problem.

Let $\phi = \bigwedge_{i=1}^{m} (a_i \vee b_i \vee c_i)$ be an instance of the 3-SAT problem, where a_i, b_i and c_i are literals over the set of variables $\{Y_1, \ldots, Y_N\}$ (a literal refers to a variable Y_j or its negation, $\bar{Y_j}$). Let C_j be the jth clause in ϕ, such that $C_j = a_j \vee b_j \vee c_j$. We define the following context-free grammar $\boldsymbol{G}_\phi$ and string to parse s_ϕ:

1. The terminals of $\boldsymbol{G}_\phi$ are the binary digits $\Sigma = \{0, 1\}$.
2. We create N nonterminals V_{Y_r}, $r \in \{1, \ldots, N\}$ and rules $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$.
3. We create N nonterminals $V_{\bar{Y}_r}$, $r \in \{1, \ldots, N\}$ and rules $V_{\bar{Y}_r} \to 0$ and $V_{\bar{Y}_r} \to 1$.
4. We create $U_{Y_r,1} \to V_{Y_r} V_{\bar{Y}_r}$ and $U_{Y_r,0} \to V_{\bar{Y}_r} V_{Y_r}$.
5. We create the rule $S_{\phi_1} \to A_1$. For each $j \in \{2, \ldots, m\}$, we create a rule $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ where S_{ϕ_j} is a new nonterminal indexed by $\phi_j \triangleq \bigwedge_{i=1}^{j} C_i$ and A_j is also a new nonterminal indexed by $j \in \{1, \ldots, m\}$.
6. Let $C_j = a_j \vee b_j \vee c_j$ be clause j in ϕ. Let $Y(a_j)$ be the variable that a_j mentions. Let (y_1, y_2, y_3) be a satisfying assignment for C_j where $y_k \in \{0, 1\}$ and is the value of $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$ respectively for $k \in \{1, 2, 3\}$. For each such clause-satisfying assignment, we add the rule:
$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (6)$$
For each A_j, we would have at most 7 rules of that form, since one rule will be logically inconsistent with $a_j \vee b_j \vee c_j$.
7. The grammar's start symbol is S_{ϕ_n}.
8. The string to parse is $s_\phi = (10)^{3m}$, i.e. $3m$ consecutive occurrences of the string 10.

A parse of the string s_ϕ using $\boldsymbol{G}_\phi$ will be used to get an assignment by setting $Y_r = 0$ if the rule $V_{Y_r} \to 0$ or $V_{\bar{Y}_r} \to 1$ are used in the derivation of the parse tree, and 1 otherwise. Notice that at this point we do not exclude "contradictions" coming from the parse tree, such as $V_{Y_3} \to 0$ used in the tree together with $V_{Y_3} \to 1$ or $V_{\bar{Y}_3} \to 0$. The following lemma gives a condition under which the assignment is consistent (so contradictions do not occur in the parse tree):

Lemma 1. *Let ϕ be an instance of the 3-SAT problem, and let $\boldsymbol{G}_\phi$ be a probabilistic CFG based on the above grammar with weights $\boldsymbol{\theta}_\phi$. If the (multiplicative) weight of the Viterbi parse of s_ϕ is 1, then the assignment extracted from the parse tree is consistent.*

Proof. Since the probability of the Viterbi parse is 1, all rules of the form $\{V_{Y_r}, V_{\bar{Y}_r}\} \to \{0, 1\}$ which appear in the parse tree have probability 1 as well. There are two possible types of inconsistencies. We show that neither exists in the Viterbi parse:

1. For any r, an appearance of both rules of the form $V_{Y_r} \to 0$ and $V_{Y_r} \to 1$ cannot occur because all rules that appear in the Viterbi parse tree have probability 1.
2. For any r, an appearance of rules of the form $V_{Y_r} \to 1$ and $V_{\bar{Y_r}} \to 1$ cannot occur, because whenever we have an appearance of the rule $V_{Y_r} \to 0$, we have an adjacent appearance of the rule $V_{\bar{Y_r}} \to 1$ (because we parse substrings of the form 10), and then again we use the fact that all rules in the parse tree have probability 1. The case of $V_{Y_r} \to 0$ and $V_{\bar{Y_r}} \to 0$ is handled analogously.

Thus, both possible inconsistencies are ruled out, resulting in a consistent assignment. □

Figure 1 gives an example of an application of the reduction.

Lemma 2. *Define ϕ, $\boldsymbol{G}_\phi$ as before. There exists $\boldsymbol{\theta}_\phi$ such that the Viterbi parse of s_ϕ is 1 if and only if ϕ is satisfiable. Moreover, the satisfying assignment is the one extracted from the parse tree with weight 1 of s_ϕ under $\boldsymbol{\theta}_\phi$.*

Proof. ($\Longrightarrow$) Assume that there is a satisfying assignment. Each clause $C_j = a_j \vee b_j \vee c_j$ is satisfied using a tuple (y_1, y_2, y_3) which assigns value for $Y(a_j)$, $Y(b_j)$ and $Y(c_j)$. This assignment corresponds the following rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (7)$$

Set its probability to 1, and set all other rules of A_j to 0. In addition, for each r, if $Y_r = y$, set the probabilities of the rules $V_{Y_r} \to y$ and $V_{\bar{Y_r}} \to 1-y$ to 1 and $V_{\bar{Y_r}} \to y$ and $V_{Y_r} \to 1 - y$ to 0. The rest of the weights for $S_{\phi_j} \to S_{\phi_{j-1}} A_j$ are set to 1. This assignment of rule probabilities results in a Viterbi parse of weight 1.

($\Longleftarrow$) Assume that the Viterbi parse has probability 1. From Lemma 1, we know that we can extract a consistent assignment from the Viterbi parse. In addition, for each clause C_j we have a rule

$$A_j \to U_{Y(a_j),y_1} U_{Y(b_j),y_2} U_{Y(c_j),y_3} \qquad (8)$$

that is assigned probability 1, for some (y_1, y_2, y_3). One can verify that (y_1, y_2, y_3) are the values of the assignment for the corresponding variables in clause C_j, and that they satisfy this clause. This means that each clause is satisfied by the assignment we extracted. □

In order to show an NP-hardness result, we need to "convert" ViterbiTrain to a decision problem. The natural way to do it, following Lemmas 1 and 2, is to state the decision problem for ViterbiTrain as "given $\boldsymbol{G}$ and $x_1, \ldots, x_n$ and $\alpha \geq 0$, is the optimized value of the objective function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z}) \geq \alpha$?" and use $\alpha = 1$ together with Lemmas 1 and 2. (Naturally, an algorithm for solving ViterbiTrain can easily be used to solve its decision problem.)

Theorem 3. *The decision version of the* ViterbiTrain *problem is NP-hard.*

5 Hardness of Approximation

A natural path of exploration following the hardness result we showed is determining whether an *approximation* of ViterbiTrain is also hard. Perhaps there is an efficient approximation algorithm for ViterbiTrain we could use instead of coordinate ascent algorithms such as Viterbi EM. Recall that such algorithms' main guarantee is identifying a local maximum; we know nothing about how far it will be from the global maximum.

We next show that approximating the objective function of ViterbiTrain with a constant factor of ρ is hard for any $\rho \in (\frac{1}{2}, 1]$ (i.e., $1/2 + \epsilon$ approximation is hard for any $\epsilon \leq 1/2$). This means that, under the P $\neq$ NP assumption, there is no efficient algorithm that, given a grammar $\boldsymbol{G}$ and a sample of sentences $x_1, \ldots, x_n$, returns $\boldsymbol{\theta}'$ and $\boldsymbol{z}'$ such that:

$$\mathcal{L}(\boldsymbol{\theta}', \boldsymbol{z}') \quad \geq \quad \rho \cdot \max_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(x_i, z_i \mid \boldsymbol{\theta}) \qquad (9)$$

We will continue to use the same reduction from §4. Let s_ϕ be the string from that reduction, and let $(\boldsymbol{\theta}, z)$ be the optimal solution for ViterbiTrain given $\boldsymbol{G}_\phi$ and s_ϕ. We first note that if $p(s_\phi, z \mid \boldsymbol{\theta}) < 1$ (implying that there is no satisfying assignment), then there must be a nonterminal which appears along with two different rules in z.

This means that we have a nonterminal $B \in \mathcal{N}$ with some rule $B \to \alpha$ that appears k times, while the nonterminal appears in the parse $r \geq k + 1$ times. Given the tree z, the $\boldsymbol{\theta}$ that maximizes the objective function is the maximum likelihood estimate (MLE) for z (counting and normalizing the rules).[2] We therefore know that the ViterbiTrain objective function, $\mathcal{L}(\boldsymbol{\theta}, z)$, is at

[2] Note that we can only make $p(z \mid \boldsymbol{\theta}, x)$ greater by using $\boldsymbol{\theta}$ to be the MLE for the derivation z.

most $\left(\frac{k}{r}\right)^k$, because it includes a factor equal to $\left(\frac{f_{B\to\alpha}(z)}{f_B(z)}\right)^{f_{B\to\alpha}(z)}$, where $f_B(z)$ is the number of times nonterminal B appears in z (hence $f_B(z) = r$) and $f_{B\to\alpha}(z)$ is the number of times $B \to \alpha$ appears in z (hence $f_{B\to\alpha}(z) = k$). For any $k \geq 1, r \geq k+1$:

$$\left(\frac{k}{r}\right)^k \leq \left(\frac{k}{k+1}\right)^k \leq \frac{1}{2} \qquad (10)$$

This means that if the value of the objective function of ViterbiTrain is not 1 using the reduction from §4, then it is at most $\frac{1}{2}$. If we had an efficient approximate algorithm with approximation coefficient $\rho > \frac{1}{2}$ (Eq. 9 holds), then in order to solve 3-SAT for formula ϕ, we could run the algorithm on $\boldsymbol{G}_\phi$ and s_ϕ and check whether the assignment to $(\boldsymbol{\theta}, z)$ that the algorithm returns satisfies ϕ or not, and return our response accordingly.

If ϕ were satisfiable, then the true maximal value of $\mathfrak{L}$ would be 1, and the approximation algorithm would return $(\boldsymbol{\theta}, z)$ such that $\mathfrak{L}(\boldsymbol{\theta}, z) \geq \rho > \frac{1}{2}$. z would have to correspond to a satisfying assignment, and in fact $p(z \mid \boldsymbol{\theta}) = 1$, because in any other case, the probability of a derivation which does not represent a satisfying assignment is smaller than $\frac{1}{2}$. If ϕ were not satisfiable, then the approximation algorithm would never return a $(\boldsymbol{\theta}, z)$ that results in a satisfying assignment (because such a $(\boldsymbol{\theta}, z)$ does not exist).

The conclusion is that an efficient algorithm for approximating the objective function of ViterbiTrain (Eq. 4) within a factor of $\frac{1}{2} + \epsilon$ is unlikely to exist. If there were such an algorithm, we could use it to solve 3-SAT using the reduction from §4.

6 Extensions of the Hardness Result

An alternative problem to Problem 1, a variant of Viterbi-training, is the following (see, for example, Klein and Manning, 2001):

Problem 3. ConditionalViterbiTrain
Input: $\boldsymbol{G}$ *context-free grammar,* $x_1, \ldots, x_n$ *training instances from* $L(\boldsymbol{G})$
Output: $\boldsymbol{\theta}$ *and* $z_1, \ldots, z_n$ *such that*

$$(\boldsymbol{\theta}, z_1, \ldots, z_n) = \operatorname*{argmax}_{\boldsymbol{\theta}, \boldsymbol{z}} \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\theta}, x_i) \quad (11)$$

Here, instead of maximizing the likelihood, we maximize the *conditional* likelihood. Note that there is a hidden assumption in this problem definition, that x_i can be parsed using the grammar $\boldsymbol{G}$. Otherwise, the quantity $p(z_i \mid \boldsymbol{\theta}, x_i)$ is not well-defined. We can extend ConditionalViterbiTrain to return $\perp$ in the case of not having a parse for one of the x_i—this can be efficiently checked using a run of a cubic-time parser on each of the strings x_i with the grammar $\boldsymbol{G}$.

An approximate technique for this problem is similar to Viterbi EM, only modifying the M-step to maximize the conditional, rather than joint, likelihood. This new M-step will not have a closed form and may require auxiliary optimization techniques like gradient ascent.

Our hardness result for ViterbiTrain applies to ConditionalViterbiTrain as well. The reason is that if $p(z, s_\phi \mid \boldsymbol{\theta}_\phi) = 1$ for a ϕ with a satisfying assignment, then $L(\boldsymbol{G}) = \{s_\phi\}$ and $D(\boldsymbol{G}) = \{z\}$. This implies that $p(z \mid \boldsymbol{\theta}_\phi, s_\phi) = 1$. If ϕ is unsatisfiable, then for the optimal $\boldsymbol{\theta}$ of ViterbiTrain we have z and z' such that $0 < p(z, s_\phi \mid \boldsymbol{\theta}_\phi) < 1$ and $0 < p(z', s_\phi \mid \boldsymbol{\theta}_\phi) < 1$, and therefore $p(z \mid \boldsymbol{\theta}_\phi, s_\phi) < 1$, which means the conditional objective function will not obtain the value 1. (Note that there always exist some parameters $\boldsymbol{\theta}_\phi$ that generate s_ϕ.) So, again, given an algorithm for ConditionalViterbiTrain, we can discern between a satisfiable formula and an unsatisfiable formula, using the reduction from §4 with the given algorithm, and identify whether the value of the objective function is 1 or strictly less than 1. We get the result that:

Theorem 4. *The decision problem of* ConditionalViterbiTrain *problem is NP-hard.*

where the decision problem of ConditionalViterbiTrain is defined analogously to the decision problem of ViterbiTrain.

We can similarly show that finding the global maximum of the marginalized likelihood:

$$\max_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \log \sum_{z} p(x_i, z \mid \boldsymbol{\theta}) \qquad (12)$$

is NP-hard. The reasoning follows. Using the reduction from before, if ϕ is satisfiable, then Eq. 12 gets value 0. If ϕ is unsatisfiable, then we would still get value 0 only if $L(\boldsymbol{G}) = \{s_\phi\}$. If $\boldsymbol{G}_\phi$ generates a single derivation for $(10)^{3m}$, then we actually do have a satisfying assignment from

Lemma 1. Otherwise (more than a single derivation), the optimal $\boldsymbol{\theta}$ would have to give fractional probabilities to rules of the form $V_{Y_r} \rightarrow \{0, 1\}$ (or $V_{\bar{Y}_r} \rightarrow \{0, 1\}$). In that case, it is no longer true that $(10)^{3m}$ is the only generated sentence, which is a contradiction.

The quantity in Eq. 12 can be maximized approximately using algorithms like EM, so this gives a hardness result for optimizing the objective function of EM for PCFGs. Day (1983) previously showed that maximizing the marginalized likelihood for hidden Markov models is NP-hard.

We note that the grammar we use for all of our results is not recursive. Therefore, we can encode this grammar as a hidden Markov model, strengthening our result from PCFGs to HMMs.[3]

7 Uniform-at-Random Initialization

In the previous sections, we showed that solving Viterbi training is hard, and therefore requires an approximation algorithm. Viterbi EM, which is an example of such algorithm, is dependent on an initialization of either $\boldsymbol{\theta}$ to start with an E-step or $\boldsymbol{z}$ to start with an M-step. In the absence of a better-informed initializer, it is reasonable to initialize $\boldsymbol{z}$ using a uniform distribution over $D(\boldsymbol{G}, x_i)$ for each i. If $D(\boldsymbol{G}, x_i)$ is finite, it can be done efficiently by setting $\boldsymbol{\theta} = \mathbf{1}$ (ignoring the normalization constraint), running the inside algorithm, and sampling from the (unnormalized) posterior given by the chart (Johnson et al., 2007). We turn next to an analysis of this initialization technique that suggests it is well-motivated.

The sketch of our result is as follows: we first give an asymptotic upper bound for the log-likelihood of derivations and sentences. This bound, which has an information-theoretic interpretation, depends on a parameter λ, which depends on the distribution from which the derivations were chosen. We then show that this bound is minimized when we pick λ such that this distribution is (conditioned on the sentence) a uniform distribution over derivations.

Let $q(x)$ be any distribution over $L(\boldsymbol{G})$ and $\boldsymbol{\theta}$ some parameters for $\boldsymbol{G}$. Let $f(z)$ be some feature function (such as the one that counts the number of appearances of a certain rule in a derivation), and then:

$$\mathbb{E}_{q,\boldsymbol{\theta}}[f] \triangleq \sum_{x \in L(\boldsymbol{G})} q(x) \sum_{z \in D(\boldsymbol{G},x)} p(z \mid \boldsymbol{\theta}, x) f(z)$$

which gives the expected value of the feature function $f(z)$ under the distribution $q(x) \times p(z \mid \boldsymbol{\theta}, x)$. We will make the following assumption about $\boldsymbol{G}$:

Condition 1. *There exists some $\boldsymbol{\theta}_I$ such that $\forall x \in L(\boldsymbol{G}), \forall z \in D(\boldsymbol{G}, x)$, $p(z \mid \boldsymbol{\theta}_I, x) = 1/|D(\boldsymbol{G}, x)|$.*

This condition is satisfied, for example, when $\boldsymbol{G}$ is in Chomsky normal form and for all $A, A' \in \mathcal{N}$, $|R(A)| = |R(A')|$. Then, if we set $\theta_{A \rightarrow \alpha} = 1/|R(A)|$, we get that all derivations of x will have the same number of rules and hence the same probability. This condition does not hold for grammars with unary cycles because $|D(\boldsymbol{G}, x)|$ may be infinite for some derivations. Such grammars are not commonly used in NLP.

Let us assume that some "correct" parameters $\boldsymbol{\theta}^*$ exist, and that our data were drawn from a distribution parametrized by $\boldsymbol{\theta}^*$. The goal of this section is to motivate the following initialization for $\boldsymbol{\theta}$, which we call UniformInit:

1. Initialize $\boldsymbol{z}$ by sampling from the uniform distribution over $D(\boldsymbol{G}, x_i)$ for each x_i.
2. Update the grammar parameters using maximum likelihood estimation.

7.1 Bounding the Objective

To show our result, we require first the following definition due to Freund et al. (1997):

Definition 5. *A distribution p_1 is within $\lambda \geq 1$ of a distribution p_2 if for every event A, we have*

$$\frac{1}{\lambda} \leq \frac{p_1(A)}{p_2(A)} \leq \lambda \tag{13}$$

For any feature function $f(z)$ and any two sets of parameters $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_1$ for $\boldsymbol{G}$ and for any marginal $q(x)$, if $p(z \mid \boldsymbol{\theta}_1, x)$ is within λ of $p(z \mid \boldsymbol{\theta}_2, x)$ for all x then:

$$\frac{\mathbb{E}_{q,\boldsymbol{\theta}_1}[f]}{\lambda} \leq \mathbb{E}_{q,\boldsymbol{\theta}_2}[f] \leq \lambda \mathbb{E}_{q,\boldsymbol{\theta}_1}[f] \tag{14}$$

Let $\boldsymbol{\theta}_0$ be a set of parameters such that we perform the following procedure in initializing Viterbi EM: first, we sample from the posterior distribution $p(z \mid \boldsymbol{\theta}_0, x)$, and then update the parameters with maximum likelihood estimate, in a regular M-step. Let λ be such that $p(z \mid \boldsymbol{\theta}_0, x)$ is within λ of $p(z \mid \boldsymbol{\theta}^*, x)$ (for all $x \in L(\boldsymbol{G})$). (Later we will show that UniformInit is a wise choice for making λ small. Note that UniformInit is equivalent to the procedure mentioned above with $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$.)

[3] We thank an anonymous reviewer for pointing this out.

Consider $\tilde{p}_n(x)$, the empirical distribution over $x_1, \ldots, x_n$. As $n \to \infty$, we have that $\tilde{p}_n(x) \to p^*(x)$, almost surely, where p^* is:

$$p^*(x) = \sum_z p^*(x, z \mid \boldsymbol{\theta}^*) \tag{15}$$

This means that as $n \to \infty$ we have $\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}}[f] \to \mathbb{E}_{p^*, \boldsymbol{\theta}}[f]$. Now, let $\boldsymbol{z}_0 = (z_{0,1}, \ldots, z_{0,n})$ be samples from $p(z \mid \boldsymbol{\theta}_0, x_i)$ for $i \in \{1, \ldots, n\}$. Then, from simple MLE computation, we know that the value

$$\begin{aligned} &\max_{\boldsymbol{\theta}'} \prod_{i=1}^{n} p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (16) \\ &= \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \end{aligned}$$

We also know that for $\boldsymbol{\theta}_0$, from the consistency of MLE, for large enough samples:

$$\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \approx \frac{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_A]} \tag{17}$$

which means that we have the following as n grows (starting from the ViterbiTrain objective with initial state $\boldsymbol{z} = \boldsymbol{z}_0$):

$$\begin{aligned} &\max_{\boldsymbol{\theta}'} \prod_{i=1}^{n} p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (18) \\ &\overset{\text{(Eq. 16)}}{=} \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{F_{A \to \alpha}(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (19) \\ &\overset{\text{(Eq. 17)}}{\approx} \prod_{(A \to \alpha) \in \mathcal{R}} \left(\frac{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{\tilde{p}_n, \boldsymbol{\theta}_0}[f_A]} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (20) \end{aligned}$$

We next use the fact that $\tilde{p}_n(x) \approx p^*(x)$ for large n, and apply Eq. 14, noting again our assumption that $p(z \mid \boldsymbol{\theta}_0, x)$ is within λ of $p(z \mid \boldsymbol{\theta}^*, x)$. We also let $B = \sum_i |z_i|$, where $|z_i|$ is the number of nodes in the derivation z_i. Note that $F_A(z_i) \leq B$. The above quantity (Eq. 20) is approximately bounded above by

$$\begin{aligned} &\prod_{(A \to \alpha) \in \mathcal{R}} \frac{1}{\lambda^{2B}} \left(\frac{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_A]} \right)^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (21) \\ &= \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0)} \qquad (22) \end{aligned}$$

Eq. 22 follows from:

$$\theta^*_{A \to \alpha} = \frac{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}^*}[f_A]} \tag{23}$$

If we continue to develop Eq. 22 and apply Eq. 17 and Eq. 23 again, we get that:

$$\begin{aligned} &\frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0)} \\ &= \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{F_{A \to \alpha}(\boldsymbol{z}_0) \cdot \frac{F_A(\boldsymbol{z}_0)}{F_A(\boldsymbol{z}_0)}} \\ &\approx \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{\frac{\mathbb{E}_{p^*, \boldsymbol{\theta}_0}[f_{A \to \alpha}]}{\mathbb{E}_{p^*, \boldsymbol{\theta}_0}[f_A]} \cdot F_A(\boldsymbol{z}_0)} \\ &\geq \frac{1}{\lambda^{2|\mathcal{R}|B}} \prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{\lambda^2 \theta^*_{A \to \alpha} F_A(\boldsymbol{z}_0)} \\ &\geq \frac{1}{\lambda^{2|\mathcal{R}|B}} \underbrace{\left(\prod_{(A \to \alpha) \in \mathcal{R}} (\theta^*_{A \to \alpha})^{n \theta^*_{A \to \alpha}} \right)}_{T(\boldsymbol{\theta}^*, n)}^{B\lambda^2/n} \qquad (24) \\ &= \left(\frac{1}{\lambda^{2|\mathcal{R}|B}} \right) T(\boldsymbol{\theta}^*, n)^{B\lambda^2/n} \qquad (25) \\ &\triangleq d(\lambda; \boldsymbol{\theta}^*, |\mathcal{R}|, B) \qquad (26) \end{aligned}$$

where Eq. 24 is the result of $F_A(\boldsymbol{z}_0) \leq B$.

For two series $\{a_n\}$ and $\{b_n\}$, let "$a_n \gtrapprox b_n$" denote that $\lim_{n \to \infty} a_n \geq \lim_{n \to \infty} b_n$. In other words, a_n is asymptotically larger than b_n. Then, if we changed the representation of the objective function of the ViterbiTrain problem to *log*-likelihood, for $\boldsymbol{\theta}'$ that maximizes Eq. 18 (with some simple algebra) we have:

$$\begin{aligned} &\frac{1}{n} \sum_{i=1}^{n} \log_2 p(x_i, z_{0,i} \mid \boldsymbol{\theta}') \qquad (27) \\ &\gtrapprox -\frac{2|\mathcal{R}|B}{n} \log_2 \lambda + \frac{B\lambda^2}{n} \left(\frac{1}{n} \log_2 T(\boldsymbol{\theta}^*, n) \right) \\ &= -\frac{2|\mathcal{R}|B}{n} \log_2 \lambda - |\mathcal{N}| \frac{B\lambda^2}{|\mathcal{N}|n} \sum_{A \in \mathcal{N}} H(\boldsymbol{\theta}^*, A) \qquad (28) \end{aligned}$$

where

$$H(\boldsymbol{\theta}^*, A) = -\sum_{(A \to \alpha) \in R(A)} \theta^*_{A \to \alpha} \log_2 \theta^*_{A \to \alpha} \tag{29}$$

is the entropy of the multinomial for nonterminal A. $H(\boldsymbol{\theta}^*, A)$ can be thought of as the minimal number of bits required to encode a choice of a rule from A, if chosen independently from the other rules. All together, the quantity $\frac{B}{|\mathcal{N}|n} \left(\sum_{A \in \mathcal{N}} H(\boldsymbol{\theta}^*, A) \right)$ is the average number of bits required to encode a tree in our sample using

$\boldsymbol{\theta}^*$, while removing dependence among all rules and assuming that each node at the tree is chosen uniformly.[4] This means that the log-likelihood, for large n, is bounded from above by a linear function of the (average) number of bits required to optimally encode n trees of total size B, while assuming independence among the rules in a tree. We note that the quantity B/n will tend toward the average size of a tree, which, under Condition 1, must be finite.

Our final approximate bound from Eq. 28 relates the choice of distribution, from which sample $\boldsymbol{z}_0$, to λ. The lower bound in Eq. 28 is a monotone-decreasing function of λ. We seek to make λ as small as possible to make the bound tight. We next show that the uniform distribution optimizes λ in that sense.

7.2 Optimizing λ

Note that the optimal choice of λ, for a single x and for candidate initializer $\boldsymbol{\theta}'$, is

$$\lambda_{\text{opt}}(x, \boldsymbol{\theta}^*; \boldsymbol{\theta}_0) \quad = \quad \sup_{z \in D(\boldsymbol{G}, x)} \frac{p(z \mid \boldsymbol{\theta}_0, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \tag{30}$$

In order to avoid degenerate cases, we will add another condition on the true model, $\boldsymbol{\theta}^*$:

Condition 2. *There exists $\tau > 0$ such that, for any $x \in L(\boldsymbol{G})$ and for any $z \in D(\boldsymbol{G}, x)$, $p(z \mid \boldsymbol{\theta}^*, x) \geq \tau$.*

This is a strong condition, forcing the cardinality of $D(\boldsymbol{G})$ to be finite, but it is not unreasonable if natural language sentences are effectively bounded in length.

Without further information about $\boldsymbol{\theta}^*$ (other than that it satisfies Condition 2), we may want to consider the worst-case scenario of possible λ, hence we seek initializer $\boldsymbol{\theta}_0$ such that

$$\Lambda(x; \boldsymbol{\theta}_0) \quad \triangleq \quad \sup_{\boldsymbol{\theta}} \lambda_{\text{opt}}(x, \boldsymbol{\theta}; \boldsymbol{\theta}_0) \tag{31}$$

is minimized. If $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$, then we have that $p(z \mid \boldsymbol{\theta}_I, x) = |D(\boldsymbol{G}, x)|^{-1} \triangleq \mu_x$. Together with Condition 2, this implies that

$$\frac{p(z \mid \boldsymbol{\theta}_I, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \quad \leq \quad \frac{\mu_x}{\tau} \tag{32}$$

and hence $\lambda_{\text{opt}}(x, \boldsymbol{\theta}^*) \leq \mu_x/\tau$ for any $\boldsymbol{\theta}^*$, hence $\Lambda(x; \boldsymbol{\theta}_I) \leq \mu_x/\tau$. However, if we choose $\boldsymbol{\theta}_0 \neq \boldsymbol{\theta}_I$, we have that $p(z' \mid \boldsymbol{\theta}_0, x) > \mu_x$ for some z', hence, for $\boldsymbol{\theta}^*$ such that it assigns probability τ on z', we have that

$$\sup_{z \in D(\boldsymbol{G}, x)} \frac{p(z \mid \boldsymbol{\theta}_0, x)}{p(z \mid \boldsymbol{\theta}^*, x)} \quad > \quad \frac{\mu_x}{\tau} \tag{33}$$

and hence $\lambda_{\text{opt}}(x, \boldsymbol{\theta}^*; \boldsymbol{\theta}') > \mu_x/\tau$, so $\Lambda(x; \boldsymbol{\theta}') > \mu_x/\tau$. So, to optimize for the worst-case scenario over true distributions with respect to λ, we are motivated to choose $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_I$ as defined in Condition 1. Indeed, UniformInit uses $\boldsymbol{\theta}_I$ to initialize the state of Viterbi EM.

We note that if $\boldsymbol{\theta}_I$ was known for a specific grammar, then we could have used it as a direct initializer. However, Condition 1 only guarantees its existence, and does not give a practical way to identify it. In general, as mentioned above, $\boldsymbol{\theta} = \mathbf{1}$ can be used to obtain a weighted CFG that satisfies $p(z \mid \boldsymbol{\theta}, x) = 1/|D(\boldsymbol{G}, x)|$. Since we require a uniform posterior distribution, the number of derivations of a fixed length is finite. This means that we can converted the weighted CFG with $\boldsymbol{\theta} = \mathbf{1}$ to a PCFG with the same posterior (Smith and Johnson, 2007), and identify the appropriate $\boldsymbol{\theta}_I$.

8 Related Work

Viterbi training is closely related to the k-means clustering problem, where the objective is to find k centroids for a given set of d-dimensional points such that the sum of distances between the points and the closest centroid is minimized. The analog for Viterbi EM for the k-means problem is the k-means clustering algorithm (Lloyd, 1982), a coordinate ascent algorithm for solving the k-means problem. It works by iterating between an E-like-step, in which each point is assigned the closest centroid, and an M-like-step, in which the centroids are set to be the center of each cluster.

"k" in k-means corresponds, in a sense, to the size of our grammar. k-means has been shown to be NP-hard both when k varies and d is fixed and when d varies and k is fixed (Aloise et al., 2009; Mahajan et al., 2009). An open problem relating to our hardness result would be whether ViterbiTrain (or ConditionalViterbiTrain) is hard even if we do not permit grammars of arbitrarily large size, or at least, constrain the number of rules that do not rewrite to terminals (in our current reduction, the

[4] We note that Grenander (1967) describes a (linear) relationship between the *derivational entropy* and $H(\boldsymbol{\theta}^*, A)$. The derivational entropy is defined as $h(\boldsymbol{\theta}^*, A) = -\sum_{x,z} p(x, z \mid \boldsymbol{\theta}^*) \log p(x, z \mid \boldsymbol{\theta}^*)$, where z ranges over trees that have nonterminal A as the root. It follows immediately from Grenander's result that $\sum_A H(\boldsymbol{\theta}^*, A) \leq \sum_A h(\boldsymbol{\theta}^*, A)$.

size of the grammar grows as the size of the 3-SAT formula grows).

On a related note to §7, Arthur and Vassilvitskii (2007) described a greedy initialization algorithm for initializing the centroids of k-means, called k-means++. They show that their initialization is $O(\log k)$-competitive; i.e., it approximates the optimal clusters assignment by a factor of $O(\log k)$. In §7.1, we showed that uniform-at-random initialization is approximately $O(|\mathcal{N}|L\lambda^2/n)$-competitive (modulo an additive constant) for CNF grammars, where n is the number of sentences, L is the total length of sentences and λ is a measure for distance between the true distribution and the uniform distribution.[5]

Many combinatorial problems in NLP involving phrase-structure trees, alignments, and dependency graphs are hard (Sima'an, 1996; Goodman, 1998; Knight, 1999; Casacuberta and de la Higuera, 2000; Lyngsø and Pederson, 2002; Udupa and Maji, 2006; McDonald and Satta, 2007; DeNero and Klein, 2008, *inter alia*). Of special relevance to this paper is Abe and Warmuth (1992), who showed that the problem of finding maximum likelihood model of probabilistic automata is hard even for a single string and an automaton with two states. Understanding the complexity of NLP problems, we believe, is crucial as we seek effective practical approximations when necessary.

9 Conclusion

We described some properties of Viterbi training for probabilistic context-free grammars. We showed that Viterbi training is NP-hard and, in fact, NP-hard to approximate. We gave motivation for uniform-at-random initialization for derivations in the Viterbi EM algorithm.

Acknowledgments

We acknowledge helpful comments by the anonymous reviewers. This research was supported by NSF grant 0915187.

References

N. Abe and M. Warmuth. 1992. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9(2–3):205–260.

S. Abney. 2007. *Semisupervised Learning for Computational Linguistics*. CRC Press.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

D. Arthur and S. Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proc. of ACM-SIAM symposium on Discrete Algorithms*.

F. Casacuberta and C. de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proc. of ICGI*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. of AAAI*.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of HLT-NAACL*.

M. Collins. 2003. Head-driven statistical models for natural language processing. *Computational Linguistics*, 29(4):589–637.

W. H. E. Day. 1983. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103.

J. DeNero and D. Klein. 2008. The complexity of phrase alignment problems. In *Proc. of ACL*.

Y. Freund, H. Seung, E. Shamir, and N. Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2–3):133–168.

S. Goldwater and M. Johnson. 2005. Bias in learning syllable structure. In *Proc. of CoNLL*.

J. Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University.

U. Grenander. 1967. Syntax-controlled probabilities. Technical report, Brown University, Division of Applied Mathematics.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *Advances in NIPS*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL*.

D. Klein and C. Manning. 2001. Natural language grammar induction using a constituent-context model. In *Advances in NIPS*.

K. Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

S. P. Lloyd. 1982. Least squares quantization in PCM. In *IEEE Transactions on Information Theory*.

R. B. Lyngsø and C. N. S. Pederson. 2002. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computing and System Science*, 65(3):545–569.

M. Mahajan, P. Nimbhorkar, and K. Varadarajan. 2009. The planar k-means problem is NP-hard. In *Proc. of International Workshop on Algorithms and Computation*.

[5] Making the assumption that the grammar is in CNF permits us to use L instead of B, since there is a linear relationship between them in that case.

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

D. McClosky, E. Charniak, and M. Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proc. of COLING-ACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

R. M. Neal and G. E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning and Graphical Models*, pages 355–368. Kluwer Academic Publishers.

K. Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *In Proc. of COLING*.

M. Sipser. 2006. *Introduction to the Theory of Computation, Second Edition*. Thomson Course Technology.

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proc. of CoNLL*.

R. Udupa and K. Maji. 2006. Computational complexity of statistical machine translation. In *Proc. of EACL*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for question answering. In *Proc. of EMNLP*.

C. Yejin and C. Cardie. 2007. Structured local training and biased potential functions for conditional random fields with application to coreference resolution. In *Proc. of HLT-NAACL*.

A Generalized-Zero-Preserving Method for Compact Encoding of Concept Lattices

Matthew Skala
School of Computer Science
University of Waterloo
mskala@cs.toronto.edu

Victoria Krakovna
János Kramár
Dept. of Mathematics
University of Toronto
{vkrakovna,jkramar}@gmail.com

Gerald Penn
Dept. of Computer Science
University of Toronto
gpenn@cs.toronto.edu

Abstract

Constructing an encoding of a concept lattice using short bit vectors allows for efficient computation of join operations on the lattice. Join is the central operation any unification-based parser must support. We extend the traditional bit vector encoding, which represents join failure using the zero vector, to count any vector with less than a fixed number of one bits as failure. This allows non-joinable elements to share bits, resulting in a smaller vector size. A constraint solver is used to construct the encoding, and a variety of techniques are employed to find near-optimal solutions and handle timeouts. An evaluation is provided comparing the extended representation of failure with traditional bit vector techniques.

1 Introduction

The use of bit vectors is almost as old as HPSG parsing itself. Since they were first suggested in the programming languages literature (Aït-Kaci et al., 1989) as a method for computing the unification of two types without table lookup, bit vectors have been attractive because of three speed advantages:

- The classical bit vector encoding uses bitwise AND to calculate type unification. This is hard to beat.
- Hash tables, the most common alternative, involve computing the Dedekind-MacNeille completion (DMC) at compile time if the input type hierarchy is not a bounded-complete partial order. That is exponential time in the worst case; most bit vector methods avoid explicitly computing it.
- With large type signatures, the table that indexes unifiable pairs of types may be so large that it pushes working parsing memory into swap. This loss of *locality of reference* costs time.

Why isn't everyone using bit vectors? For the most part, the reason is their size. The classical encoding given by Aït-Kaci et al. (1989) is at least as large as the number of meet-irreducible types, which in the parlance of HPSG type signatures is the number of unary-branching types plus the number of maximally specific types. For the English Resource Grammar (ERG) (Copestake and Flickinger, 2000), these are 314 and 2474 respectively. While some systems use them nonetheless (PET (Callmeier, 2000) does, as a very notable exception), it is clear that the size of these codes is a source of concern.

Again, it has been so since the very beginning: Aït-Kaci et al. (1989) devoted several pages to a discussion of how to "modularize" type codes, which typically achieves a smaller code in exchange for a larger-time operation than bitwise AND as the implementation of type unification. However, in this and later work on the subject (e.g. (Fall, 1996)), one constant has been that we know our unification has failed when the implementation returns the zero vector. *Zero preservation* (Mellish, 1991; Mellish, 1992), i.e., detecting a type unification failure, is just as important as obtaining the right answer quickly when it succeeds.

The approach of the present paper borrows from recent statistical machine translation research, which addresses the problem of efficiently representing large-scale language models using a mathematical construction called a *Bloom filter* (Talbot and Osborne, 2007). The approach is best combined with modularization in order to further reduce the size of the codes, but its novelty lies in

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1512–1521,
Uppsala, Sweden, 11-16 July 2010.

the observation that counting the number of one bits in an integer is implemented in the basic instruction sets of many CPUs. The question then arises whether smaller codes would be obtained by relaxing zero preservation so that *any* resulting vector with at most λ bits is interpreted as failure, with $\lambda \geq 1$.

Penn (2002) generalized join-preserving encodings of partial orders to the case where more than one code can be used to represent the same object, but the focus there was on codes arising from successful unifications; there was still only one representative for failure. To our knowledge, the present paper is the first generalization of zero preservation in CL or any other application domain of partial order encodings.

We note at the outset that we are not using Bloom filters as such, but rather a derandomized encoding scheme that shares with Bloom filters the essential insight that λ can be greater than zero without adverse consequences for the required algebraic properties of the encoding. Deterministic variants of Bloom filters may in turn prove to be of some value in language modelling.

1.1 Notation and definitions

A partial order $\langle X, \sqsubseteq \rangle$ consists of a set X and a reflexive, antisymmetric, and transitive binary relation $\sqsubseteq$. We use $u \sqcup v$ to denote the unique least upper bound or *join* of $u, v \in X$, if one exists, and $u \sqcap v$ for the greatest lower bound or *meet*. If we need a second partial order, we use $\preceq$ for its order relation and $\curlyvee$ for its join operation. We are especially interested in a class of partial orders called *meet semilattices*, in which every pair of elements has a unique meet. In a meet semilattice, the join of two elements is unique when it exists at all, and there is a unique globally least element $\bot$ ("*bottom*").

A *successor* of an element $u \in X$ is an element $v \neq u \in X$ such that $u \sqsubseteq v$ and there is no $w \in X$ with $w \neq u, w \neq v$, and $u \sqsubseteq w \sqsubseteq v$, i.e., v follows u in X with no other elements in between. A *maximal* element has no successor. A *meet irreducible* element is an element $u \in X$ such that for any $v, w \in X$, if $u = v \sqcap w$ then $u = v$ or $u = w$. A meet irreducible has at most one successor.

Given two partial orders $\langle X, \sqsubseteq \rangle$ and $\langle Y, \preceq \rangle$, an *embedding* of X into Y is a pair of functions $f : X \to Y$ and $g : (Y \times Y) \to \{0, 1\}$, which may have some of the following properties for all $u, v \in X$:

$$u \sqsubseteq v \Rightarrow f(u) \preceq f(v) \quad (1)$$

$$\mathit{defined}(u \sqcup v) \Rightarrow g(f(u), f(v)) = 1 \quad (2)$$

$$\neg \mathit{defined}(u \sqcup v) \Rightarrow g(f(u), f(v)) = 0 \quad (3)$$

$$u \sqcup v = w \Leftrightarrow f(u) \curlyvee f(v) = f(w) \quad (4)$$

With property (1), the embedding is said to preserve order; with property (2), it preserves success; with property (3), it preserves failure; and with property (4), it preserves joins.

2 Bit-vector encoding

Intuitively, taking the join of two types in a type hierarchy is like taking the intersection of two sets. Types often represent sets of possible values, and the type represented by the join really does represent the intersection of the sets that formed the input. So it seems natural to embed a partial order of types $\langle X, \sqsubseteq \rangle$ into a partial order (in fact, a lattice) of sets $\langle Y, \preceq \rangle$, where Y is the power set of some set Z, and $\preceq$ is the superset relation $\supseteq$. Then join $\curlyvee$ is simply set intersection $\cap$. The embedding function g, which indicates whether a join exists, can be naturally defined by $g(f(u), f(v)) = 0$ if and only if $f(u) \cap f(v) = \emptyset$. It remains to choose the underlying set Z and embedding function f.

Aït-Kaci et al. (1989) developed what has become the standard technique of this type. They set Z to be the set of all meet irreducible elements in X; and $f(u) = \{v \in Z | v \sqsupseteq u\}$, that is, the meet irreducible elements greater than or equal to u. The resulting embedding preserves order, success, failure, and joins. If Z is chosen to be the maximal elements of X instead, then join preservation is lost but the embedding still preserves order, success, and failure. The sets can be represented efficiently by vectors of bits. We hope to minimize the size of the largest set $f(\bot)$, which determines the vector length.

It follows from the work of Markowsky (1980) that the construction of Aït-Kaci et al. is optimal among encodings that use sets with intersection for meet and empty set for failure: with Y defined as the power set of some set Z, $\sqsubseteq$ as $\supseteq$, $\sqcup$ as $\cap$, and $g(f(u), f(v)) = 0$ if and only if $f(u) \cap f(v) = \emptyset$, then the smallest Z that will preserve order, success, failure, and joins is the set of all meet irreducible elements of X. No shorter bit vectors are possible.

We construct shorter bit vectors by modifying the definition of g, so that the minimality results

no longer apply. In the following discussion we present first an intuitive and then a technical description of our approach.

2.1 Intuition from Bloom filters

Vectors generated by the above construction tend to be quite sparse, or if not sparse, at least boring. Consider a meet semilattice containing only the bottom element $\perp$ and n maximal elements all incomparable to each other. Then each bit vector would consist of either all ones, or all zeroes except for a single one. We would thus be spending n bits to represent a choice among $n + 1$ alternatives, which should fit into a logarithmic number of bits. The meet semilattices that occur in practice are more complicated than this example, but they tend to contain things like it as a substructure. With the traditional bit vector construction, each of the maximal elements consumes its own bit, even though those bits are highly correlated.

The well-known technique called Bloom filtering (Bloom, 1970) addresses a similar issue. There, it is desired to store a large array of bits subject to two considerations. First, most of the bits are zeroes. Second, we are willing to accept a small proportion of one-sided errors, where every query that should correctly return one does so, but some queries that should correctly return zero might actually return one instead.

The solution proposed by Bloom and widely used in the decades since is to map the entries in the large bit array pseudorandomly (by means of a hash function) into the entries of a small bit array. To store a one bit we find its hashed location and store it there. If we query a bit for which the answer should be zero but it happens to have the same hashed location as another query with the answer one, then we return a one and that is one of our tolerated errors.

To reduce the error rate we can elaborate the construction further: with some fixed k, we use k hash functions to map each bit in the large array to several locations in the small one. Figure 1 illustrates the technique with $k = 3$. Each bit has three hashed locations. On a query, we check all three; they must all contain ones for the query to return a one. There will be many collisions of individual hashed locations, as shown; but the chances are good that when we query a bit we did not intend to store in the filter, at least one of its hashed locations will still be empty, and so the query will

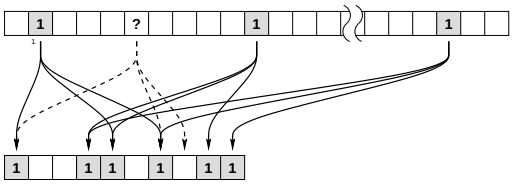


Figure 1: A Bloom filter

return zero. Bloom describes how to calculate the optimal value of k, and the necessary length of the hashed array, to achieve any desired bound on the error rate. In general, the hashed array can be much smaller than the original unhashed array (Bloom, 1970).

Classical Bloom filtering applied to the sparse vectors of the embedding would create some percentage of incorrect join results, which would then have to be handled by other techniques. Our work described here combines the idea of using k hash functions to reduce the error rate, with perfect hashes designed in a precomputation step to bring the error rate to zero.

2.2 Modified failure detection

In the traditional bit vector construction, types map to sets, join is computed by intersection of sets, and the empty set corresponds to failure (where no join exists). Following the lead of Bloom filters, we change the embedding function $g(f(u), f(v))$ to be 0 if and only if $|f(u) \cap f(v)| \leq \lambda$ for some constant λ. With $\lambda = 0$ this is the same as before. Choosing greater values of λ allows us to re-use set elements in different parts of the type hierarchy while still avoiding collisions.

Figure 2 shows an example meet semilattice. In the traditional construction, to preserve joins we must assign one bit to each of the meet-irreducible elements $\{d, e, f, g, h, i, j, k, l, m\}$, for a total of ten bits. But we can use eight bits and still preserve joins by setting $g(f(u), f(v)) = 0$ if and only if $|f(u) \cap f(v)| \leq \lambda = 1$, and f as follows.

$$
\begin{array}{cc}
f(\perp) = \{1,2,3,4,5,6,7,8\} & \\
f(a) = \{1,2,3,4,5\} & \\
f(b) = \{1,6,7,8\} & f(c) = \{1,2,3\} \\
f(d) = \{2,3,4,5\} & f(e) = \{1,6\} \\
f(f) = \{1,7\} & f(g) = \{1,8\} \\
f(h) = \{6,7\} & f(i) = \{6,8\} \\
f(j) = \{1,2\} & f(k) = \{1,3\} \\
f(l) = \{2,3\} & f(m) = \{2,3,4\}
\end{array}
\tag{5}
$$

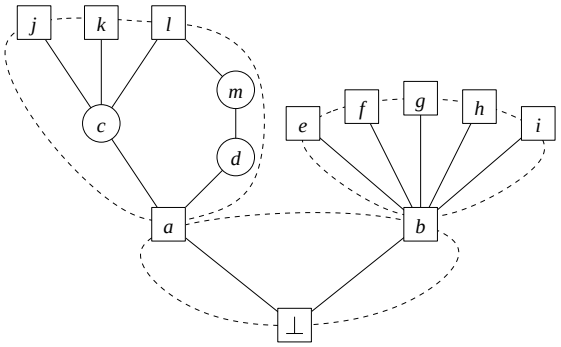


Figure 2: An example meet semilattice; $\perp$ is the most general type.

As a more general example, consider the very simple meet semilattice consisting of just a least element $\perp$ with n maximal elements incomparable to each other. For a given λ we can represent this in b bits by choosing the smallest b such that $\binom{b}{\lambda+1} \geq n$ and assigning each maximal element a distinct choice of the bits. With optimal choice of λ, b is logarithmic in n.

2.3 Modules

As Aït-Kaci et al. (1989) described, partial orders encountered in practice often resemble trees. Both their technique and ours are at a disadvantage when applied to large trees; in particular, if the bottom of the partial order has successors which are not joinable with each other, then those will be assigned large sets with little overlap, and bits in the vectors will tend to be wasted.

To avoid wasting bits, we examine the partial order X in a precomputation step to find the *modules*, which are the smallest upward-closed subsets of X such that for any $x \in X$, if x has at least two joinable successors, then x is in a module. This is similar to ALE's definition of module (Penn, 1999), but not the same. The definition of Aït-Kaci et al. (1989) also differs from ours. Under our definition, every module has a unique least element, and not every type is in a module. For instance, in Figure 2, the only module has a as its least element. In the ERG's type hierarchy, there are 11 modules, with sizes ranging from 10 to 1998 types.

To find the join of two types in the same module, we find the intersection of their encodings and check whether it is of size greater than λ. If the types belong to two distinct modules, there is no join. For the remaining cases, where at least one of the types lacks a module, we observe that the module bottoms and non-module types form a tree, and the join can be computed in that tree. If x is a type in the module whose bottom is y, and z has no module, then $x \sqcup z = y \sqcup z$ unless $y \sqcup z = y$ in which case $x \sqcup z = x$; so it only remains to compute joins within the tree. Our implementation does that by table lookup. More sophisticated approaches could be appropriate on larger trees.

3 Set programming

Ideally, we would like to have an efficient algorithm for finding the best possible encoding of any given meet semilattice. The encoding can be represented as a collection of sets of integers (representing bit indices that contain ones), and an optimal encoding is the collection of sets whose overall union is smallest subject to the constraint that the collection forms an encoding at all. This combinatorial optimization problem is a form of *set programming*; and set programming problems are widely studied. We begin by defining the form of set programming we will use.

Definition 1 *Choose set variables $S_1, S_2, \ldots, S_n$ to minimize $b = |\bigcup_{i=1}^{n} S_i|$ subject to some constraints of the forms $|S_i| \geq r_i$, $S_i \subseteq S_j$, $S_i \not\supseteq S_j$, $|S_i \cap S_j| \leq \lambda$, and $S_i \cap S_j = S_k$. The constant λ is the same for all constraints. Set elements may be arbitrary, but we generally assume they are the integers $\{1 \ldots b\}$ for convenience.*

The reduction of partial order representation to set programming is clear: we create a set variable for every type, force the maximal types' sets to contain at least $\lambda + 1$ elements, and then use subset to enforce that every type is a superset of all its successors (preserving order and success). We limit the maximum intersection of incomparable types to preserve failure. To preserve joins, if that property is desired, we add a constraint $S_i \not\supseteq S_j$ for every pair of types $x_i \not\sqsubseteq x_j$ and one of the form $S_i \cap S_j = S_k$ for every x_i, x_j, x_k such that $x_i \sqcup x_j = x_k$..

Given a constraint satisfaction problem like this one, we can ask two questions: is there a *feasible* solution, assigning values to the variables so all constraints are satisfied; and if so what is the *optimal* solution, producing the best value of the objective while remaining feasible? In our problem, there is always a feasible solution we can find by the *generalized Aït-Kaci et al. construction* (GAK), which consists of assigning λ bits

shared among all types; adding enough unshared new bits to maximal elements to satisfy cardinality constraints; adding one new bit to each non-maximal meet irreducible type; and propagating all the bits down the hierarchy to satisfy the subset constraints. Since the GAK solution is feasible, it provides a useful upper bound on the result of the set programming.

Ongoing research on set programming has produced a variety of software tools for solving these problems. However, at first blush our instances are much too large for readily-available set programming tools. Grammars like ERG contain thousands of types. We use binary constraints between every pair of types, for a total of millions of constraints—and these are variables and constraints over a domain of sets, not integers or reals. General-purpose set programming software cannot handle such instances.

3.1 Simplifying the instances

First of all, we only use minimum cardinality constraints $|S_i| \geq r_i$ for maximal types; and every $r_i \geq \lambda + 1$. Given a feasible bit assignment for a maximal type with more than r_i elements in its set S_i, we can always remove elements until it has exactly r_i elements, without violating the other constraints. As a result, instead of using constraints $|S_i| \geq r_i$ we can use constraints $|S_i| = r_i$. Doing so reduces the search space.

Subset is transitive; so if we have constraints $S_i \subseteq S_j$ and $S_j \subseteq S_k$, then $S_i \subseteq S_k$ is implied and we need not specify it as a constraint. Similarly, if we have $S_i \subseteq S_j$ and $S_i \nsubseteq S_k$, then we have $S_j \nsubseteq S_k$. Furthermore, if S_i and S_j have maximum intersection λ, then any subset of S_i also has maximum intersection λ with any subset of S_k, and we need not specify those constraints either.

Now, let a *choke-vertex* in the partial order $\langle X, \sqsubseteq \rangle$ be an element $u \in X$ such that for every $v, w \in X$ where v is a successor of w and $u \sqsubseteq v$, we have $u \sqsubseteq w$. That is, any chain of successors from elements not after u to elements after u, must pass through u. Figure 2 shows choke-vertices as squares. We call these choke-vertices by analogy with the graph theoretic concept of cut-vertices in the Hasse diagram of the partial order; but note that some vertices (like j and k) can be choke-vertices without being cut-vertices, and some vertices (like c) can be cut-vertices without being choke-vertices. Maximal and minimal elements are always choke-vertices.

Choke-vertices are important because the optimal bit assignment for elements after a choke-vertex u is almost independent of the bit assignment elsewhere in the partial order. Removing the redundant constraints means there are no constraints between elements after u and elements before, or incomparable with, u. All constraints across u must involve u directly. As a result, we can solve a smaller instance consisting of u and everything after it, to find the minimal number of bits r_u for representing u. Then we solve the rest of the problem with a constraint $|S_u| = r_u$, excluding all partial order elements after u, and then combine the two solutions with any arbitrary bijection between the set elements assigned to u in each solution. Assuming optimal solutions to both sub-problems, the result is an optimal solution to the original problem.

3.2 Splitting into components

If we cut the partial order at every choke-vertex, we reduce the huge and impractical encoding problem to a collection of smaller ones. The cutting expresses the original partial order as a tree of *components*, each of which corresponds to a set programming instance. Components are shown by the dashed lines in Figure 2. We can find an optimal encoding for the entire partial order by optimally encoding the components, starting with the leaves of that tree and working our way back to the root.

The division into components creates a collection of set programming instances with a wide range of sizes and difficulty; we examine each instance and choose appropriate techniques for each one. Table 1 summarizes the rules used to solve an instance, and shows the number of times each rule was applied in a typical run with the modules extracted from ERG, a ten-minute timeout, and each λ from 0 to 10.

In many simple cases, GAK is provably optimal. These include when $\lambda = 0$ regardless of the structure of the component; when the component consists of a bottom and zero, one, or two non-joinable successors; and when there is one element (a *top*) greater than all other elements in the component. We can easily recognize these cases and apply GAK to them.

Another important special case is when the

Condition	Succ.	Fail.	Method
$\lambda = 0$	216		GAK (optimal)
$\exists$ top	510		GAK (optimal)
2 successors	850		GAK (optimal)
3 or 4 successors	70		exponential variable
only ULs	420		b-choose-$(\lambda+1)$ special case
before UL removal	251	59	`ic_sets`
after UL removal	9	50	`ic_sets`
remaining	50		GAK

Table 1: Rules for solving an instance in the ERG

component consists of a bottom and some number k of pairwise non-joinable successors, and the successors all have required cardinality $\lambda + 1$. Then the optimal encoding comes from finding the smallest b such that $\binom{b}{\lambda+1}$ is at least k, and giving each successor a distinct combination of the b bits.

3.3 Removing unary leaves

For components that do not have one of the special forms described above, it becomes necessary to solve the set programming problem. Some of our instances are small enough to apply constraint solving software directly; but for larger instances, we have one more technique to bring them into the tractable range.

Definition 2 *A* unary leaf *(UL) is an element x in a partial order $\langle X, \sqsubseteq \rangle$ such that x is maximal and x is the successor of exactly one other element.*

ULs are special because their set programming constraints always take a particular form: if x is a UL and a successor of y, then the constraints on its set S_x are exactly that $|S_x| = \lambda + 1$, $S_x \subseteq S_y$, and S_x has intersection of size at most λ with the set for any other successor of y. Other constraints disappear by the simplifications described earlier.

Furthermore, ULs occur frequently in the partial orders we consider in practice; and by increasing the number of sets in an instance, they have a disproportionate effect on the difficulty of solving the set programming problem. We therefore implement a special solution process for instances containing ULs: we remove them all, solve the resulting instance, and then add them back one at a time while attempting to increase the overall number of elements as little as possible.

This process of removing ULs, solving, and adding them back in, may in general produce suboptimal solutions, so we use it only when the solver cannot find a solution on the full-sized problem. In practical experiments, the solver generally either produces an optimal or very nearly optimal solution within a time limit on the order of ten minutes; or fails to produce a feasible solution at all, even with a much longer limit. Testing whether it finds a solution is then a useful way to determine whether UL removal is worthwhile.

Recall that in an instance consisting of k ULs and a bottom, an optimal solution consists of finding the smallest b such that $\binom{b}{\lambda+1}$ is at least k; that is the number of bits for the bottom, and we can choose any k distinct subsets of size $\lambda + 1$ for the ULs. Augmenting an existing solution to include additional ULs involves a similar calculation.

To add a UL x as the successor of an element y without increasing the total number of bits, we must find a choice of $\lambda + 1$ of the bits already assigned to y, sharing at most λ bits with any of y's other successors. Those successors are in general sets of arbitrary size, but all that matters for assigning x is how many subsets of size $\lambda + 1$ they already cover. The UL can use any such subset not covered by an existing successor of y. Our algorithm counts the subsets already covered, and compares that with the number of choices of $\lambda+1$ bits from the bits assigned to y. If enough choices remain, we use them; otherwise, we add bits until there are enough choices.

3.4 Solving

For instances with a small number of sets and relatively large number of elements in the sets, we use an *exponential variable* solver. This encodes the set programming problem into integer programming. For each element $x \in \{1, 2, \ldots, b\}$, let $c(x) = \{i | x \in S_i\}$; that is, $c(x)$ represents the indices of all the sets in the problem that contain the element x. There are $2^n - 1$ possible values of $c(x)$, because each element must be in at least one set. We create an integer variable for each of those values. Each element is counted once, so the sum of the integer variables is b. The constraints translate into simple inequalities on sums of the variables; and the system of constraints can be solved with standard integer programming techniques. After solving the integer programming problem we can then assign elements arbitrarily

to the appropriate combinations of sets.

Where applicable, the exponential variable approach works well, because it breaks all the symmetries between set elements. It also continues to function well even when the sets are large, since nothing in the problem directly grows when we increase b. The wide domains of the variables may be advantageous for some integer programming solvers as well. However, it creates an integer programming problem of size exponential in the number of sets. As a result, it is only applicable to instances with a very few set variables.

For more general set programming instances, we feed the instance directly into a solver designed for such problems. We used the ECL^{i}PS^e logic programming system (Cisco Systems, 2008), which offers several set programming solvers as libraries, and settled on the ic_sets library. This is a straightforward set programming solver based on containment bounds. We extended the solver by adding a lightweight not-subset constraint, and customized heuristics for variable and value selection designed to guide the solver to a feasible solution as soon as possible. We choose variables near the top of the instance first, and prefer to assign values that share exactly λ bits with existing assigned values. We also do limited symmetry breaking, in that whenever we assign a bit not shared with any current assignment, the choice of bit is arbitrary so we assume it must be the lowest-index bit. That symmetry breaking speeds up the search significantly.

The present work is primarily on the benefits of nonzero λ, and so a detailed study of general set programming techniques would be inappropriate; but we made informal tests of several other set-programming solvers. We had hoped that a solver using containment-lexicographic hybrid bounds as described by Sadler and Gervet (Sadler and Gervet, 2008) would offer good performance, and chose the ECL^{i}PS^e framework partly to gain access to its ic_hybrid_sets implementation of such bounds. In practice, however, ic_hybrid_sets gave consistently worse performance than ic_sets (typically by an approximate factor of two). It appears that in intuitive terms, the lexicographic bounds rarely narrowed the domains of variables much until the variables were almost entirely labelled anyway, at which point containment bounds were almost as good; and meanwhile the increased overhead of maintaining the extra bounds slowed down the entire process to more than compensate for the improved propagation. We also evaluated the Cardinal solver included in ECL^{i}PS^e, which offers stronger propagation of cardinality information; it lacked other needed features and seemed no more efficient than ic_sets. Among these three solvers, the improvements associated with our custom variable and value heuristics greatly outweighed the baseline differences between the solvers; and the differences were in optimization time rather than quality of the returned solutions.

Solvers with available source code were preferred for ease of customization, and free solvers were preferred for economy, but a license for ILOG CPLEX (IBM, 2008) was available and we tried using it with the natural encoding of sets as vectors of binary variables. It solved small instances to optimality in time comparable to that of ECL^{i}PS^e. However, for medium to large instances, CPLEX proved impractical. An instance with n sets of up to b bits, dense with pairwise constraints like subset and maximum intersection, requires $\Theta(n^2b)$ variables when encoded into integer programming in the natural way. CPLEX stores a copy of the relaxed problem, with significant bookkeeping information per variable, for every node in the search tree. It is capable of storing most of the tree in compressed form on disk, but in our larger instances even a single node is too large; CPLEX exhausts memory while loading its input. The ECL^{i}PS^e solver also stores each set variable in a data structure that increases linearly with the number of elements, so that the size of the problem as stored by ECL^{i}PS^e is also $\Theta(n^2b)$; but the constant for ECL^{i}PS^e appears to be much smaller, and its search algorithm stores only incremental updates (with nodes per set instead of per element) on a stack as it explores the tree. As a result, the ECL^{i}PS^e solver can process much larger instances than CPLEX without exhausting memory.

Encoding into SAT would allow use of the sophisticated solvers available for that problem. Unfortunately, cardinality constraints are notoriously difficult to encode in Boolean logic. The obvious encoding of our problem into CNFSAT would require $O(n^2b\lambda)$ clauses and variables. Encodings into Boolean variables with richer constraints than CNFSAT (we tried, for instance, the SICStus Prolog clp(FD) implementation (Carlsson et al., 1997)) generally exhausted memory on much smaller instances than those handled by the set-

Module	n	b_0	λ	b_λ
`mrs_min`	10	7	0	7
`conj`	13	8	1	7
`list`	27	15	1	11
`local_min`	27	21	1	10
`cat_min`	30	17	1	14
`individual`	33	15	0	15
`head_min`	247	55	0	55
`*sort*`	247	129	3	107
`synsem_min`	612	255	0	255
`sign_min`	1025	489	3	357
`mod_relation`	1998	1749	6	284
entire ERG	4305	2788	140	985

Table 2: Best encodings of the ERG and its modules: n is number of types, b_0 is vector length with $\lambda = 0$, and λ is parameter that gives the shortest vector length b_λ.

variable solvers, while offering no improvement in speed.

4 Evaluation

Table 2 shows the size of our smallest encodings to date for the entire ERG without modularization, and for each of its modules. These were found by running the optimization process of the previous section on Intel Xeon servers with a timeout of 30 minutes for each invocation of the solver (which may occur several times per module). Under those conditions, some modules take a long time to optimize—as much as two hours per tested value of λ for `sign_min`. The Xeon's hyperthreading feature makes reproducibility of timing results difficult, but we found that results almost never improved with additional time allowance beyond the first few seconds in any case, so the practical effect of the timing variations should be minimal.

These results show some significant improvements in vector length for the larger modules. However, they do not reveal the entire story. In particular, the apparent superiority of $\lambda = 0$ for the `synsem_min` module should not be taken as indicating that no higher λ could be better: rather, that module includes a very difficult set programming instance on which the solver failed and fell back to GAK. For the even larger modules, nonzero λ proved helpful despite solver failures, because of the bits saved by UL removal. UL removal is clearly a significant advantage, but only for the modules where the solver is failing anyway. One important lesson seems to be that further work on set programming solvers would be beneficial: any future more capable set programming solver could be applied to the unsolved instances and would be expected to save more bits.

Encoding	length	time	space
Lookup table	n/a	140	72496
Modular, best λ	0–357	321	203
Modular, $\lambda = 0$	0–1749	747	579
Non-mod, $\lambda = 0$	2788	4651	1530
Non-mod, $\lambda = 1$	1243	2224	706
Non-mod, $\lambda = 2$	1140	2008	656
Non-mod, $\lambda = 9$	1069	1981	622
Non-mod, $\lambda = 140$	985	3018	572

Table 3: Query performance. Vector length in bits, time in milliseconds, space in Kbytes.

Table 3 and Figure 3 show the performance of the join query with various encodings. These results are from a simple implementation in C that tests all ordered pairs of types for joinability. As well as testing the non-modular ERG encoding for different values of λ, we tested the modularized encoding with $\lambda = 0$ for all modules (to show the effect of modularization alone) and with λ chosen per-module to give the shortest vectors. For comparison, we also tested a simple lookup table. The same implementation sufficed for all these tests, by means of putting all types in one module for the non-modular bit vectors or no types in any module for the pure lookup table. The times shown are milliseconds of user CPU time to test all join tests (roughly 18.5 million of them), on a non-hyperthreading Intel Pentium 4 with a clock speed of 2.66GHz and 1G of RAM, running Linux. Space consumption shown is the total amount of dynamically-allocated memory used to store the vectors and lookup table.

The non-modular encoding with $\lambda = 0$ is the basic encoding of Aït-Kaci et al. (1989). As Table 3 shows, we achieved more than a factor of two improvement from that, in both time and vector length, just by setting $\lambda = 1$. Larger values offered further small improvements in length up to $\lambda = 140$, which gave the minimum vector length of 985. That is a shallow minimum; both $\lambda = 120$ and $\lambda = 160$ gave vector lengths of 986, and the length slowly increased with greater λ.

However, the fastest bit-count on this architec-

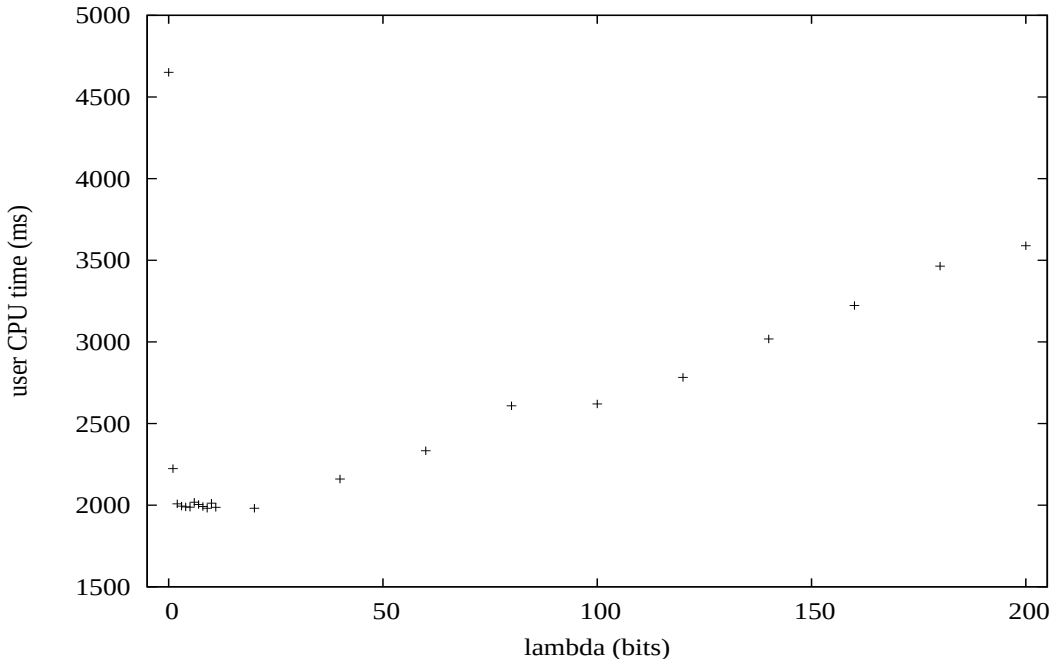


Figure 3: Query performance for the ERG without modularization.

ture, using a technique first published by Wegner (1960), requires time increasing with the number of nonzero bits it counts; and a similar effect would appear on a word-by-word basis even if we used a constant-time per-word count. As a result, there is a time cost associated with using larger λ, so that the fastest value is not necessarily the one that gives the shortest vectors. In our experiments, $\lambda = 9$ gave the fastest joins for the non-modular encoding of the ERG. As shown in Figure 3, all small nonzero λ gave very similar times.

Modularization helps a lot, both with $\lambda = 0$, and when we choose the optimal λ per module. Here, too, the use of optimal λ improves both time and space by more than a factor of two. Our best bit-vector encoding, the modularized one with per-module optimal λ, is only a little less than half the speed of the lookup table; and this test favours the lookup table by giving it a full word for every entry (no time spent shifting and masking bits) and testing the pairs in a simple two-level loop (almost purely sequential access).

5 Conclusion

We have described a generalization of conventional bit vector concept lattice encoding techniques to the case where all vectors with λ or fewer one bits represent failure; traditional encodings are the case $\lambda = 0$. Increasing λ can reduce the overall storage space and improve speed.

A good encoding requires a kind of perfect hash, the design of which maps naturally to constraint programming over sets of integers. We have described a practical framework for solving the instances of constraint programming thus created, in which we can apply existing or future constraint solvers to the subproblems for which they are best suited; and a technique for modularizing practical type hierarchies to get better value from the bit vector encodings. We have evaluated the resulting encodings on the ERG's type system, and examined the performance of the associated unification test. Modularization, and the use of nonzero λ, each independently provide significant savings in both time and vector length.

The modified failure detection concept suggests several directions for future work, including evaluation of the new encodings in the context of a large-scale HPSG parser; incorporation of further developments in constraint solvers; and the possibility of approximate encodings that would permit one-sided errors as in traditional Bloom filtering.

References

Hassan Aït-Kaci, Robert S. Boyer, Patrick Lincoln, and Roger Nasr. 1989. Efficient implementation of lattice operations. *ACM Transactions on Programming Languages and Systems*, 11(1):115–146, January.

Burton H. Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July.

Ulrich Callmeier. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107.

Mats Carlsson, Greger Ottosson, and Björn Carlson. 1997. An open-ended finite domain constraint solver. In H. Glaser, P. Hartel, and H. Kucken, editors, *Programming Languages: Implementations, Logics, and Programming*, volume 1292 of *Lecture Notes in Computer Science*, pages 191–206. Springer-Verlag, September.

Cisco Systems. 2008. ECL^{i}PS^e 6.0. Computer software. Online `http://eclipse-clp.org/`.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second Conference on Language Resources and Evaluation (LREC 2000)*.

Andrew Fall. 1996. *Reasoning with Taxonomies*. Ph.D. thesis, Simon Fraser University.

IBM. 2008. ILOG CPLEX 11. Computer software.

George Markowsky. 1980. The representation of posets and lattices by sets. *Algebra Universalis*, 11(1):173–192.

Chris Mellish. 1991. Graph-encodable description spaces. Technical report, University of Edinburgh Department of Artificial Intelligence. DYANA Deliverable R3.2B.

Chris Mellish. 1992. Term-encodable description spaces. In D.R. Brough, editor, *Logic Programming: New Frontiers*, pages 189–207. Kluwer.

Gerald Penn. 1999. An optimized prolog encoding of typed feature structures. In D. De Schreye, editor, *Logic programming: proceedings of the 1999 International Conference on Logic Programming (ICLP)*, pages 124–138.

Gerald Penn. 2002. Generalized encoding of description spaces and its application to typed feature structures. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, pages 64–71.

Andrew Sadler and Carmen Gervet. 2008. Enhancing set constraint solvers with lexicographic bounds. *Journal of Heuristics*, 14(1).

David Talbot and Miles Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476.

Peter Wegner. 1960. A technique for counting ones in a binary computer. *Communications of the ACM*, 3(5):322.

Knowledge-rich Word Sense Disambiguation Rivaling Supervised Systems

Simone Paolo Ponzetto
Department of Computational Linguistics
Heidelberg University
ponzetto@cl.uni-heidelberg.de

Roberto Navigli
Dipartimento di Informatica
Sapienza Università di Roma
navigli@di.uniroma1.it

Abstract

One of the main obstacles to high-performance Word Sense Disambiguation (WSD) is the knowledge acquisition bottleneck. In this paper, we present a methodology to automatically extend WordNet with large amounts of semantic relations from an encyclopedic resource, namely Wikipedia. We show that, when provided with a vast amount of high-quality semantic relations, simple knowledge-lean disambiguation algorithms compete with state-of-the-art supervised WSD systems in a coarse-grained all-words setting and outperform them on gold-standard domain-specific datasets.

1 Introduction

Knowledge lies at the core of Word Sense Disambiguation (WSD), the task of computationally identifying the meanings of words in context (Navigli, 2009b). In the recent years, two main approaches have been studied that rely on a fixed sense inventory, i.e., supervised and knowledge-based methods. In order to achieve high performance, supervised approaches require large training sets where instances (target words in context) are hand-annotated with the most appropriate word senses. Producing this kind of knowledge is extremely costly: at a throughput of one sense annotation per minute (Edmonds, 2000) and tagging one thousand examples per word, dozens of person-years would be required for enabling a supervised classifier to disambiguate all the words in the English lexicon with high accuracy. In contrast, knowledge-based approaches exploit the information contained in wide-coverage lexical resources, such as WordNet (Fellbaum, 1998). However, it has been demonstrated that the amount of lexical and semantic information contained in such resources is typically insufficient for high-performance WSD (Cuadros and Rigau, 2006). Several methods have been proposed to automatically extend existing resources (cf. Section 2) and it has been shown that highly-interconnected semantic networks have a great impact on WSD (Navigli and Lapata, 2010). However, to date, the real potential of knowledge-rich WSD systems has been shown only in the presence of either a large manually-developed extension of WordNet (Navigli and Velardi, 2005) or sophisticated WSD algorithms (Agirre et al., 2009).

The contributions of this paper are two-fold. First, we relieve the knowledge acquisition bottleneck by developing a methodology to extend WordNet with millions of semantic relations. The relations are harvested from an encyclopedic resource, namely Wikipedia. Wikipedia pages are automatically associated with WordNet senses, and topical, semantic associative relations from Wikipedia are transferred to WordNet, thus producing a much richer lexical resource. Second, two simple knowledge-based algorithms that exploit our extended WordNet are applied to standard WSD datasets. The results show that the integration of vast amounts of semantic relations in knowledge-based systems yields performance competitive with state-of-the-art supervised approaches on open-text WSD. In addition, we support previous findings from Agirre et al. (2009) that in a domain-specific WSD scenario knowledge-based systems perform better than supervised ones, and we show that, given enough knowledge, simple algorithms perform better than more sophisticated ones.

2 Related Work

In the last three decades, a large body of work has been presented that concerns the development of automatic methods for the enrichment of existing resources such as WordNet. These in-

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1522–1531,
Uppsala, Sweden, 11-16 July 2010.

clude proposals to extract semantic information from dictionaries (e.g. Chodorow et al. (1985) and Rigau et al. (1998)), approaches using lexico-syntactic patterns (Hearst, 1992; Cimiano et al., 2004; Girju et al., 2006), heuristic methods based on lexical and semantic regularities (Harabagiu et al., 1999), taxonomy-based ontologization (Pennacchiotti and Pantel, 2006; Snow et al., 2006). Other approaches include the extraction of semantic preferences from sense-annotated (Agirre and Martinez, 2001) and raw corpora (McCarthy and Carroll, 2003), as well as the disambiguation of dictionary glosses based on cyclic graph patterns (Navigli, 2009a). Other works rely on the disambiguation of collocations, either obtained from specialized learner's dictionaries (Navigli and Velardi, 2005) or extracted by means of statistical techniques (Cuadros and Rigau, 2008), e.g. based on the method proposed by Agirre and de Lacalle (2004). But while most of these methods represent state-of-the-art proposals for enriching lexical and taxonomic resources, none concentrates on augmenting WordNet with associative semantic relations for many domains on a very large scale. To overcome this limitation, we exploit Wikipedia, a collaboratively generated Web encyclopedia.

The use of collaborative contributions from volunteers has been previously shown to be beneficial in the Open Mind Word Expert project (Chklovski and Mihalcea, 2002). However, its current status indicates that the project remains a mainly academic attempt. In contrast, due to its low entrance barrier and vast user base, Wikipedia provides large amounts of information at practically no cost. Previous work aimed at transforming its content into a knowledge base includes open-domain relation extraction (Wu and Weld, 2007), the acquisition of taxonomic (Ponzetto and Strube, 2007a; Suchanek et al., 2008; Wu and Weld, 2008) and other semantic relations (Nastase and Strube, 2008), as well as lexical reference rules (Shnarch et al., 2009). Applications using the knowledge contained in Wikipedia include, among others, text categorization (Gabrilovich and Markovitch, 2006), computing semantic similarity of texts (Gabrilovich and Markovitch, 2007; Ponzetto and Strube, 2007b; Milne and Witten, 2008a), coreference resolution (Ponzetto and Strube, 2007b), multi-document summarization (Nastase, 2008), and text generation (Sauper and Barzilay, 2009).

In our work we follow this line of research and show that knowledge harvested from Wikipedia can be used effectively to improve the performance of a WSD system. Our proposal builds on previous insights from Bunescu and Paşca (2006) and Mihalcea (2007) that pages in Wikipedia can be taken as word senses. Mihalcea (2007) manually maps Wikipedia pages to WordNet senses to perform lexical-sample WSD. We extend her proposal in three important ways: (1) we *fully automatize* the mapping between Wikipedia pages and WordNet senses; (2) we use the mappings to *enrich an existing resource*, i.e. WordNet, rather than annotating text with sense labels; (3) we deploy the knowledge encoded by this mapping to perform *unrestricted* WSD, rather than apply it to a lexical sample setting.

Knowledge from Wikipedia is injected into a WSD system by means of a mapping to WordNet. Previous efforts aimed at automatically linking Wikipedia to WordNet include full use of the first WordNet sense heuristic (Suchanek et al., 2008), a graph-based mapping of Wikipedia categories to WordNet synsets (Ponzetto and Navigli, 2009), a model based on vector spaces (Ruiz-Casado et al., 2005) and a supervised approach using keyword extraction (Reiter et al., 2008). These latter methods rely only on text overlap techniques and neither they take advantage of the input from Wikipedia being semi-structured, e.g. hyperlinked, nor they propose a high-performing probabilistic formulation of the mapping problem, a task to which we turn in the next section.

3 Extending WordNet

Our approach consists of two main phases: first, a mapping is automatically established between Wikipedia pages and WordNet senses; second, the relations connecting Wikipedia pages are transferred to WordNet. As a result, an extended version of WordNet is produced, that we call WordNet++. We present the two resources used in our methodology in Section 3.1. Sections 3.2 and 3.3 illustrate the two phases of our approach.

3.1 Knowledge Resources

WordNet. Being the most widely used computational lexicon of English in Natural Language Processing, WordNet is an essential resource for WSD. A concept in WordNet is represented as a synonym set, or *synset*, i.e. the set of words which share a common meaning. For instance, the con-

cept of soda drink is expressed as:

{ pop^2_n, soda^2_n, soda pop^1_n, soda water^2_n, tonic^2_n }

where each word's subscripts and superscripts indicate their parts of speech (e.g. n stands for noun) and sense number[1], respectively. For each synset, WordNet provides a textual definition, or gloss. For example, the gloss of the above synset is: "a sweet drink containing carbonated water and flavoring".

Wikipedia. Our second resource, Wikipedia, is a collaborative Web encyclopedia composed of pages[2]. A Wikipedia page (henceforth, Wikipage) presents the knowledge about a specific concept (e.g. SODA (SOFT DRINK)) or named entity (e.g. FOOD STANDARDS AGENCY). The page typically contains hypertext linked to other relevant Wikipages. For instance, SODA (SOFT DRINK) is linked to COLA, FLAVORED WATER, LEMONADE, and many others. The title of a Wikipage (e.g. SODA (SOFT DRINK)) is composed of the lemma of the concept defined (e.g. soda) plus an optional label in parentheses which specifies its meaning in case the lemma is ambiguous (e.g. SOFT DRINK VS. SODIUM CARBONATE). Finally, some Wikipages are redirections to other pages, e.g. SODA (SODIUM CARBONATE) redirects to SODIUM CARBONATE.

3.2 Mapping Wikipedia to WordNet

During the first phase of our methodology we aim to establish links between Wikipages and WordNet senses. Formally, given the entire set of pages $Senses_{\text{Wiki}}$ and WordNet senses $Senses_{\text{WN}}$, we aim to acquire a mapping:

$$\mu : Senses_{\text{Wiki}} \rightarrow Senses_{\text{WN}},$$

such that, for each Wikipage $w \in Senses_{\text{Wiki}}$:

$$\mu(w) = \begin{cases} s \in Senses_{\text{WN}}(w) & \text{if a link can be established,} \\ \epsilon & \text{otherwise,} \end{cases}$$

where $Senses_{\text{WN}}(w)$ is the set of senses of the *lemma* of w in WordNet. For example, if our mapping methodology linked SODA (SOFT DRINK) to the corresponding WordNet sense soda^2_n, we would have μ(SODA (SOFT DRINK)) = soda^2_n.

In order to establish a mapping between the two resources, we first identify different kinds of disambiguation contexts for Wikipages (Section 3.2.1) and WordNet senses (Section 3.2.2). Next, we intersect these contexts to perform the mapping (see Section 3.2.3).

3.2.1 Disambiguation Context of a Wikipage

Given a target Wikipage w which we aim to map to a WordNet sense of w, we use the following information as a disambiguation context:

- **Sense labels**: e.g. given the page SODA (SOFT DRINK), the words soft and drink are added to the disambiguation context.
- **Links**: the titles' lemmas of the pages linked from the Wikipage w (outgoing links). For instance, the links in the Wikipage SODA (SOFT DRINK) include soda, lemonade, sugar, etc.
- **Categories**: Wikipages are classified according to one or more categories, which represent meta-information used to categorize them. For instance, the Wikipage SODA (SOFT DRINK) is categorized as SOFT DRINKS. Since many categories are very specific and do not appear in WordNet (e.g., SWEDISH WRITERS or SCIENTISTS WHO COMMITTED SUICIDE), we use the lemmas of their syntactic heads as disambiguation context (i.e. writer and scientist). To this end, we use the category heads provided by Ponzetto and Navigli (2009).

Given a Wikipage w, we define its disambiguation context $Ctx(w)$ as the set of words obtained from some or all of the three sources above.

3.2.2 Disambiguation Context of a WordNet Sense

Given a WordNet sense s and its synset S, we use the following information as disambiguation context to provide evidence for a potential link in our mapping μ:

- **Synonymy**: all synonyms of s in synset S. For instance, given the synset of soda^2_n, all its synonyms are included in the context (that is, tonic, soda pop, pop, etc.).

[1] We use WordNet version 3.0. We use word senses to unambiguously denote the corresponding synsets (e.g. plane^1_n for { airplane^1_n, aeroplane^1_n, plane^1_n }).

[2] `http://download.wikipedia.org`. We use the English Wikipedia database dump from November 3, 2009, which includes 3,083,466 articles. Throughout this paper, we use Sans Serif for words, SMALL CAPS for Wikipedia pages and CAPITALS for Wikipedia categories.

- **Hypernymy/Hyponymy**: all synonyms in the synsets H such that H is either a hypernym (i.e., a generalization) or a hyponym (i.e., a specialization) of S. For example, given soda^2_n, we include the words from its hypernym { soft drink^1_n }.

- **Sisterhood**: words from the sisters of S. A sister synset S' is such that S and S' have a common direct hypernym. For example, given soda^2_n, it can be found that bitter lemon^1_n and soda^2_n are sisters. Thus the words bitter and lemon are included in the disambiguation context of s.

- **Gloss**: the set of lemmas of the content words occurring within the gloss of s. For instance, given $s = \text{soda}^2_n$, defined as "a sweet drink containing carbonated water and flavoring", we add to the disambiguation context of s the following lemmas: sweet, drink, contain, carbonated, water, flavoring.

Given a WordNet sense s, we define its disambiguation context $Ctx(s)$ as the set of words obtained from some or all of the four sources above.

3.2.3 Mapping Algorithm

In order to link each Wikipedia page to a WordNet sense, we developed a novel algorithm, whose pseudocode is shown in Algorithm 1. The following steps are performed:

- Initially (lines 1-2), our mapping μ is empty, i.e. it links each Wikipage w to ϵ.

- For each Wikipage w whose lemma is monosemous both in Wikipedia and WordNet (i.e. $|Senses_{\text{Wiki}}(w)| = |Senses_{\text{WN}}(w)| = 1$) we map w to its only WordNet sense w^1_n (lines 3-5).

- Finally, for each remaining Wikipage w for which no mapping was previously found (i.e., $\mu(w) = \epsilon$, line 7), we do the following:
  - lines 8-10: for each Wikipage d which is a redirection to w, for which a mapping was previously found (i.e. $\mu(d) \neq \epsilon$, that is, d is monosemous in both Wikipedia and WordNet) and such that it maps to a sense $\mu(d)$ in a synset S that also contains a sense of w, we map w to the corresponding sense in S.
  - lines 11-14: if a Wikipage w has not been linked yet, we assign the most likely sense to w based on the maximization of the conditional probabilities $p(s|w)$ over the senses $s \in Senses_{\text{WN}}(w)$ (no mapping is established if a tie occurs, line 13).

Algorithm 1 The mapping algorithm

Input: $Senses_{\text{Wiki}}$, $Senses_{\text{WN}}$
Output: a mapping $\mu : Senses_{\text{Wiki}} \rightarrow Senses_{\text{WN}}$

1: **for each** $w \in Senses_{\text{Wiki}}$
2: $\quad \mu(w) := \epsilon$
3: **for each** $w \in Senses_{\text{Wiki}}$
4: $\quad$ **if** $|Senses_{\text{Wiki}}(w)| = |Senses_{\text{WN}}(w)| = 1$ **then**
5: $\quad\quad \mu(w) := w^1_n$
6: **for each** $w \in Senses_{\text{Wiki}}$
7: $\quad$ **if** $\mu(w) = \epsilon$ **then**
8: $\quad\quad$ **for each** $d \in Senses_{\text{Wiki}}$ s.t. d redirects to w
9: $\quad\quad\quad$ **if** $\mu(d) \neq \epsilon$ and $\mu(d)$ is in a synset of w **then**
10: $\quad\quad\quad\quad \mu(w) :=$ sense of w in synset of $\mu(d)$; **break**
11: **for each** $w \in Senses_{\text{Wiki}}$
12: $\quad$ **if** $\mu(w) = \epsilon$ **then**
13: $\quad\quad$ **if** no tie occurs **then**
14: $\quad\quad\quad \mu(w) := \underset{s \in Senses_{\text{WN}}(w)}{\text{argmax}} \; p(s|w)$
15: **return** μ

As a result of the execution of the algorithm, the mapping μ is returned (line 15). At the heart of the mapping algorithm lies the calculation of the conditional probability $p(s|w)$ of selecting the WordNet sense s given the Wikipage w. The sense s which maximizes this probability can be obtained as follows:

$$\mu(w) = \underset{s \in Senses_{\text{WN}}(w)}{\text{argmax}} \; p(s|w) = \underset{s}{\text{argmax}} \frac{p(s,w)}{p(w)} = \underset{s}{\text{argmax}} \, p(s,w)$$

The latter formula is obtained by observing that $p(w)$ does not influence our maximization, as it is a constant independent of s. As a result, the most appropriate sense s is determined by maximizing the joint probability $p(s,w)$ of sense s and page w. We estimate $p(s,w)$ as:

$$p(s,w) = \frac{score(s,w)}{\sum_{\substack{s' \in Senses_{\text{WN}}(w), \\ w' \in Senses_{\text{Wiki}}(w)}} score(s',w')},$$

where $score(s,w) = |Ctx(s) \cap Ctx(w)| + 1$ (we add 1 as a smoothing factor). Thus, in our algorithm we determine the best sense s by computing the intersection of the disambiguation contexts of s and w, and normalizing by the scores summed over all senses of w in Wikipedia and WordNet.

3.2.4 Example

We illustrate the execution of our mapping algorithm by way of an example. Let us focus on the

Wikipage SODA (SOFT DRINK). The word soda is polysemous both in Wikipedia and WordNet, thus lines 3–5 of the algorithm do not concern this Wikipage. Lines 6–14 aim to find a mapping μ(SODA (SOFT DRINK)) to an appropriate WordNet sense of the word. First, we check whether a redirection exists to SODA (SOFT DRINK) that was previously disambiguated (lines 8–10). Next, we construct the disambiguation context for the Wikipage by including words from its label, links and categories (cf. Section 3.2.1). The context includes, among others, the following words: soft, drink, cola, sugar. We now construct the disambiguation context for the two WordNet senses of soda (cf. Section 3.2.2), namely the sodium carbonate (#1) and the drink (#2) senses. To do so, we include words from their synsets, hypernyms, hyponyms, sisters, and glosses. The context for soda_n^1 includes: salt, acetate, chlorate, benzoate. The context for soda_n^2 contains instead: soft, drink, cola, bitter, etc. The sense with the largest intersection is #2, so the following mapping is established: μ(SODA (SOFT DRINK)) = soda_n^2.

3.3 Transferring Semantic Relations

The output of the algorithm presented in the previous section is a mapping between Wikipages and WordNet senses (that is, implicitly, synsets). Our insight is to use this alignment to enable the *transfer of semantic relations from Wikipedia to WordNet*. In fact, given a Wikipage w we can collect all Wikipedia links occurring in that page. For any such link from w to w', if the two Wikipages are mapped to WordNet senses (i.e., $\mu(w) \neq \epsilon$ and $\mu(w') \neq \epsilon$), we can transfer the corresponding edge $(\mu(w), \mu(w'))$ to WordNet. Note that $\mu(w)$ and $\mu(w')$ are noun senses, as Wikipages describe nominal concepts or named entities. We refer to this extended resource as WordNet++.

For instance, consider the Wikipage SODA (SOFT DRINK). This page contains, among others, a link to the Wikipage SYRUP. Assuming μ(SODA (SODA DRINK)) = soda_n^2 and μ(SYRUP) = syrup_n^1, we can add the corresponding semantic relation (soda_n^2, syrup_n^1) to WordNet[3].

Thus, WordNet++ represents an extension of WordNet which includes semantic associative relations between synsets. These are originally found in Wikipedia and then integrated into WordNet by means of our mapping. In turn, WordNet++ represents the English-only subset of a larger multilingual resource, BabelNet (Navigli and Ponzetto, 2010), where lexicalizations of the synsets are harvested for many languages using the so-called Wikipedia inter-language links and applying a machine translation system.

[3]Note that such relations are *unlabeled*. However, for our purposes this has no impact, since our algorithms do not distinguish between *is-a* and other kinds of relations in the lexical knowledge base (cf. Section 4.2).

4 Experiments

We perform two sets of experiments: we first evaluate the intrinsic quality of our mapping (Section 4.1) and then quantify the impact of WordNet++ for coarse-grained (Section 4.2) and domain-specific WSD (Section 4.3).

4.1 Evaluation of the Mapping

Experimental setting. We first conducted an evaluation of the mapping quality. To create a gold standard for evaluation, we started from the set of all lemmas contained both in WordNet and Wikipedia: the intersection between the two resources includes 80,295 lemmas which correspond to 105,797 WordNet senses and 199,735 Wikipedia pages. The average polysemy is 1.3 and 2.5 for WordNet senses and Wikipages, respectively (2.8 and 4.7 when excluding monosemous words). We selected a random sample of 1,000 Wikipages and asked an annotator with previous experience in lexicographic annotation to provide the correct WordNet sense for each page title (an empty sense label was given if no correct mapping was possible). 505 non-empty mappings were found, i.e. Wikipedia pages with a corresponding WordNet sense. In order to quantify the quality of the annotations and the difficulty of the task, a second annotator sense tagged a subset of 200 pages from the original sample. We computed the inter-annotator agreement using the kappa coefficient (Carletta, 1996) and found out that our annotators achieved an agreement coefficient κ of 0.9, indicating almost perfect agreement.

Table 1 summarizes the performance of our disambiguation algorithm against the manually annotated dataset. Evaluation is performed in terms of standard measures of precision (the ratio of correct sense labels to the non-empty labels output by the mapping algorithm), recall (the ratio of correct sense labels to the total of non-empty labels in the gold standard) and F_1-measure ($\frac{2PR}{P+R}$). We also calculate accuracy, which accounts for

	P	R	F_1	A
Structure	**82.2**	68.1	74.5	81.1
Gloss	81.1	64.2	71.7	78.8
Structure + Gloss	81.9	**77.5**	**79.6**	**84.4**
MFS BL	24.3	47.8	32.2	24.3
Random BL	23.8	46.8	31.6	23.9

Table 1: Performance of the mapping algorithm.

empty sense labels (that is, calculated on all 1,000 test instances). As baseline we use the most frequent WordNet sense (MFS), as well as a random sense assignment. We evaluate the mapping methodology described in Section 3.2 against different disambiguation contexts for the WordNet senses (cf. Section 3.2.2), i.e. structure-based (including synonymy, hypernymy/hyponymy and sisterhood), gloss-derived evidence, and a combination of the two. As disambiguation context of a Wikipage (Section 3.2.1) we use all information available, i.e. sense labels, links and categories[4].

Results and discussion. The results show that our method improves on the baseline by a large margin and that higher performance can be achieved by using more disambiguation information. That is, using a richer disambiguation context helps to better choose the most appropriate WordNet sense for a Wikipedia page. The combination of structural and gloss information attains a slight variation in terms of precision (-0.3% and $+0.8\%$ compared to Structure and Gloss respectively), but a significantly high increase in recall ($+9.4\%$ and $+13.3\%$). This implies that the different disambiguation contexts only partially overlap and, when used separately, each produces different mappings with a similar level of precision. In the joint approach, the harmonic mean of precision and recall, i.e. F_1, is in fact 5 and 8 points higher than when separately using structural and gloss information, respectively.

As for the baselines, the most frequent sense is just 0.6% and 0.4% above the random baseline in terms of F_1 and accuracy, respectively. A χ^2 test reveals in fact no statistically significant difference at $p < 0.05$. This is related to the random distribution of senses in our dataset and the Wikipedia unbiased coverage of WordNet senses. So selecting the most frequent sense rather than any other sense for each target page represents a choice as arbitrary as picking a sense at random.

The final mapping contains 81,533 pairs of Wikipages and word senses they map to, covering 55.7% of the noun senses in WordNet.

Using our best performing mapping we are able to extend WordNet with 1,902,859 semantic edges: of these, 97.93% are deemed novel, i.e. no direct edge could previously be found between the synsets. In addition, we performed a stricter evaluation of the novelty of our relations by checking whether these can still be found indirectly by searching for a connecting path between the two synsets of interest. Here we found that 91.3%, 87.2% and 78.9% of the relations are novel to WordNet when performing a graph search of maximum depth of 2, 3 and 4, respectively.

4.2 Coarse-grained WSD

Experimental setting. We extrinsically evaluate the impact of WordNet++ on the Semeval-2007 coarse-grained all-words WSD task (Navigli et al., 2007). Performing experiments in a coarse-grained setting is a natural choice for several reasons: first, it has been argued that the fine granularity of WordNet is one of the main obstacles to accurate WSD (cf. the discussion in Navigli (2009b)); second, the meanings of Wikipedia pages are intuitively coarser than those in WordNet[5]. For instance, mapping TRAVEL to the first or the second sense in WordNet is an arbitrary choice, as the Wikipage refers to both senses. Finally, given their different nature, WordNet and Wikipedia do not fully overlap. Accordingly, we expect the transfer of semantic relations from Wikipedia to WordNet to have sometimes the side effect to penalize some fine-grained senses of a word.

We experiment with two simple knowledge-based algorithms that are set to perform coarse-grained WSD on a sentence-by-sentence basis:

- **Simplified Extended Lesk (ExtLesk):** The first algorithm is a simplified version of the Lesk

[4]We leave out the evaluation of different contexts for a Wikipage for the sake of brevity. During prototyping we found that the best results were given by using the largest context available, as reported in Table 1.

[5]Note that our polysemy rates from Section 4.1 also include Wikipages whose lemma is contained in WordNet, but which have out-of-domain meanings, i.e. encyclopedic entries referring to specialized named entities such as e.g., DISCOVERY (SPACE SHUTTLE) or FIELD ARTILLERY (MAGAZINE). We computed the polysemy rate for a random sample of 20 polysemous words by manually removing these NEs and found that Wikipedia's polysemy rate is indeed lower than that of WordNet – i.e. average polysemy of 2.1 vs. 2.8.

algorithm (Lesk, 1986), that performs WSD based on the overlap between the context surrounding the target word to be disambiguated and the definitions of its candidate senses (Kilgarriff and Rosenzweig, 2000). Given a target word w, this method assigns to w the sense whose gloss has the highest overlap (i.e. most words in common) with the context of w, namely the set of content words co-occurring with it in a pre-defined window (a sentence in our case). Due to the limited context provided by the WordNet glosses, we follow Banerjee and Pedersen (2003) and expand the gloss of each sense s to include words from the glosses of those synsets in a semantic relation with s. These include all WordNet synsets which are directly connected to s, either by means of the semantic pointers found in WordNet or through the unlabeled links found in WordNet++.

- **Degree Centrality (Degree):** The second algorithm is a graph-based approach that relies on the notion of vertex degree (Navigli and Lapata, 2010). Starting from each sense s of the target word, it performs a depth-first search (DFS) of the WordNet(++) graph and collects all the paths connecting s to senses of other words in context. As a result, a sentence graph is produced. A maximum search depth is established to limit the size of this graph. The sense of the target word with the highest vertex degree is selected. We follow Navigli and Lapata (2010) and run Degree in a weakly supervised setting where the system attempts no sense assignment if the highest degree score is below a certain (empirically estimated) threshold. The optimal threshold and maximum search depth are estimated by maximizing Degree's F_1 on a development set of 1,000 randomly chosen noun instances from the SemCor corpus (Miller et al., 1993). Experiments on the development dataset using Degree on WordNet++ revealed a performance far lower than expected. Error analysis showed that many instances were incorrectly disambiguated, due to the noise from weak semantic links, e.g. the links from SODA (SOFT DRINK) to EUROPE or AUSTRALIA. Accordingly, in order to improve the disambiguation performance, we developed a filter to rule out weak semantic relations from WordNet++. Given a WordNet++ edge $(\mu(w), \mu(w'))$ where w and w' are both Wikipages and w links to w', we first collect all words from the category labels of w and w' into two bags of words. We remove stopwords and lemmatize the remaining words. We then compute the degree of overlap between the two sets of categories as the number of words in common between the two bags of words, normalized in the $[0, 1]$ interval. We finally retain the link for the DFS if such score is above an empirically determined threshold. The optimal value for this category overlap threshold was again estimated by maximizing Degree's F_1 on the development set. The final graph used by Degree consists of WordNet, together with 152,944 relations from our semantic relation enrichment method (cf. Section 3.3).

Resource	Algorithm	Nouns only		
		P	R	F_1
WordNet	ExtLesk	83.6	57.7	68.3
	Degree	86.3	65.5	74.5
Wikipedia	ExtLesk	82.3	64.1	72.0
	Degree	**96.2**	40.1	57.4
WordNet++	ExtLesk	82.7	69.2	75.4
	Degree	87.3	72.7	**79.4**
	MFS BL	77.4	**77.4**	77.4
	Random BL	63.5	63.5	63.5

Table 2: Performance on Semeval-2007 coarse-grained all-words WSD (nouns only subset).

Results and discussion. We report our results in terms of precision, recall and F_1-measure on the Semeval-2007 coarse-grained all-words dataset (Navigli et al., 2007). We first evaluated ExtLesk and Degree using three different resources: (1) WordNet only; (2) Wikipedia only, i.e. only those relations harvested from the links found within Wikipedia pages; (3) their union, i.e. WordNet++. In Table 2 we report the results on nouns only. As common practice, we compare with random sense assignment and the most frequent sense (MFS) from SemCor as baselines. Enriching WordNet with encyclopedic relations from Wikipedia yields a consistent improvement against using WordNet (+7.1% and +4.9% F_1 for ExtLesk and Degree) or Wikipedia (+3.4% and +22.0%) alone. The best results are obtained by using Degree with WordNet++. The better performance of Wikipedia against WordNet when using ExtLesk (+3.7%) highlights the quality of the relations extracted. However, no such improvement is found with De-

Algorithm	Nouns only P/R/F_1	All words P/R/F_1
ExtLesk	81.0	79.1
Degree	**85.5**	**81.7**
SUSSX-FR	81.1	77.0
TreeMatch	N/A	73.6
NUS-PT	82.3	**82.5**
SSI	**84.1**	**83.2**
MFS BL	77.4	78.9
Random BL	63.5	62.7

Table 3: Performance on Semeval-2007 coarse-grained all-words WSD with MFS as a back-off strategy when no sense assignment is attempted.

gree, due to its lower recall. Interestingly, Degree on WordNet++ beats the MFS baseline, which is notably a difficult competitor for unsupervised and knowledge-lean systems.

We finally compare our two algorithms using WordNet++ with state-of-the-art WSD systems, namely the best unsupervised (Koeling and McCarthy, 2007, SUSSX-FR) and supervised (Chan et al., 2007, NUS-PT) systems participating in the Semeval-2007 coarse-grained all-words task. We also compare with SSI (Navigli and Velardi, 2005) – a knowledge-based system that participated out of competition – and the unsupervised proposal from Chen et al. (2009, TreeMatch). Table 3 shows the results for nouns (1,108) and all words (2,269 words): we use the MFS as a back-off strategy when no sense assignment is attempted. Degree with WordNet++ achieves the best performance in the literature[6]. On the noun-only subset of the data, its performance is comparable with SSI and significantly better than the best supervised and unsupervised systems (+3.2% and +4.4% F_1 against NUS-PT and SUSSX-FR). On the entire dataset, it outperforms SUSSX-FR and TreeMatch (+4.7% and +8.1%) and its recall is not statistically different from that of SSI and NUS-PT. This result is particularly interesting, given that WordNet++ is extended only with relations between nominals, and, in contrast to SSI, it does not rely on a costly annotation effort to engineer the set of semantic relations. Last but not least, we achieve state-of-the-art performance with a much simpler algorithm that is based on the notion of vertex degree in a graph.

[6]The differences between the results in bold in each column of the table are not statistically significant at $p < 0.05$.

Algorithm	Sports P/R/F_1	Finance P/R/F_1
k-NN †	30.3	43.4
Static PR †	20.1	39.6
Personalized PR †	35.6	46.9
ExtLesk	40.1	45.6
Degree	**42.0**	**47.8**
MFS BL	19.6	37.1
Random BL	19.5	19.6

Table 4: Performance on the Sports and Finance sections of the dataset from Koeling et al. (2005): † indicates results from Agirre et al. (2009).

4.3 Domain WSD

The main strength of Wikipedia is to provide wide coverage for many specific domains. Accordingly, on the Semeval dataset our system achieves the best performance on a domain-specific text, namely d004, a document on computer science where we achieve 82.9% F_1 (+6.8% when compared with the best supervised system, namely NUS-PT). To test whether our performance on the Semeval dataset is an artifact of the data, i.e. d004 coming from Wikipedia itself, we evaluated our system on the Sports and Finance sections of the domain corpora from Koeling et al. (2005). In Table 4 we report our results on these datasets and compare them with Personalized PageRank, the state-of-the-art system from Agirre et al. (2009)[7], as well as Static PageRank and a k-NN supervised WSD system trained on SemCor.

The results we obtain on the two domains with our best configuration (Degree using WordNet++) outperform by a large margin k-NN, thus supporting the findings from Agirre et al. (2009) that knowledge-based systems exhibit a more robust performance than their supervised alternatives when evaluated across different domains. In addition, our system achieves better results than Static and Personalized PageRank, indicating that competitive disambiguation performance can still be achieved by a less sophisticated knowledge-based WSD algorithm when provided with a rich amount of high-quality knowledge. Finally, the results show that WordNet++ enables competitive performance also in a fine-grained domain setting.

[7]We compare only with those system configurations performing token-based WSD, i.e. disambiguating each instance of a target word separately, since our aim is not to perform type-based disambiguation.

5 Conclusions

In this paper, we have presented a large-scale method for the automatic enrichment of a computational lexicon with encyclopedic relational knowledge[8]. Our experiments show that the large amount of knowledge injected into WordNet is of high quality and, more importantly, it enables simple knowledge-based WSD systems to perform as well as the highest-performing supervised ones in a coarse-grained setting and to outperform them on domain-specific text. Thus, our results go one step beyond previous findings (Cuadros and Rigau, 2006; Agirre et al., 2009; Navigli and Lapata, 2010) and prove that knowledge-rich disambiguation is a competitive alternative to supervised systems, even when relying on a simple algorithm. We note, however, that the present contribution does not show *which* knowledge-rich algorithm performs best with WordNet++. In fact, more sophisticated approaches, such as Personalized PageRank (Agirre and Soroa, 2009), could be still applied to yield even higher performance. We leave such exploration to future work. Moreover, while the mapping has been used to enrich WordNet with a large amount of semantic edges, the method can be reversed and applied to the encyclopedic resource itself, that is Wikipedia, to perform disambiguation with the corresponding sense inventory (cf. the task of wikification proposed by Mihalcea and Csomai (2007) and Milne and Witten (2008b)). In this paper, we focused on English Word Sense Disambiguation. However, since WordNet++ is part of a multilingual semantic network (Navigli and Ponzetto, 2010), we plan to explore the impact of this knowledge in a multilingual setting.

[8]The resulting resource, WordNet++, is freely available at `http://lcl.uniroma1.it/wordnetplusplus` for research purposes.

References

Eneko Agirre and Oier Lopez de Lacalle. 2004. Publicly available topic signatures for all WordNet nominal senses. In *Proc. of LREC '04*.

Eneko Agirre and David Martinez. 2001. Learning class-to-class selectional preferences. In *Proceedings of CoNLL-01*, pages 15–22.

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proc. of EACL-09*, pages 33–41.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2009. Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *Proc. of IJCAI-09*, pages 1501–1506.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlap as a measure of semantic relatedness. In *Proc. of IJCAI-03*, pages 805–810.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, pages 9–16.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007. NUS-ML: Exploiting parallel texts for Word Sense Disambiguation in the English all-words tasks. In *Proc. of SemEval-2007*, pages 253–256.

Ping Chen, Wei Ding, Chris Bowes, and David Brown. 2009. A fully unsupervised Word Sense Disambiguation method using dependency knowledge. In *Proc. of NAACL-HLT-09*, pages 28–36.

Tim Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with Open Mind Word Expert. In *Proceedings of the ACL-02 Workshop on WSD: Recent Successes and Future Directions at ACL-02*.

Martin Chodorow, Roy Byrd, and George E. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proc. of ACL-85*, pages 299–304.

Philipp Cimiano, Siegfried Handschuh, and Steffen Staab. 2004. Towards the self-annotating Web. In *Proc. of WWW-04*, pages 462–471.

Montse Cuadros and German Rigau. 2006. Quality assessment of large scale knowledge resources. In *Proc. of EMNLP-06*, pages 534–541.

Montse Cuadros and German Rigau. 2008. KnowNet: building a large net of knowledge from the Web. In *Proc. of COLING-08*, pages 161–168.

Philip Edmonds. 2000. Designing a task for SENSEVAL-2. Technical report, University of Brighton, U.K.

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

Evgeniy Gabrilovich and Shaul Markovitch. 2006. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proc. of AAAI-06*, pages 1301–1306.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, pages 1606–1611.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135.

Sanda M. Harabagiu, George A. Miller, and Dan I. Moldovan. 1999. WordNet 2 – a morphologically and semantically enhanced resource. In *Proceedings of the SIGLEX99 Workshop on Standardizing Lexical Resources*, pages 1–8.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING-92*, pages 539–545.

Adam Kilgarriff and Joseph Rosenzweig. 2000. Framework and results for English SENSEVAL. *Computers and the Humanities*, 34(1-2).

Rob Koeling and Diana McCarthy. 2007. Sussx: WSD using automatically acquired predominant senses. In *Proc. of SemEval-2007*, pages 314–317.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proc. of HLT-EMNLP-05*, pages 419–426.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual Conference on Systems Documentation,* Toronto, Ontario, Canada, pages 24–26.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Rada Mihalcea and Andras Csomai. 2007. Wikify! Linking documents to encyclopedic knowledge. In *Proc. of CIKM-07*, pages 233–242.

Rada Mihalcea. 2007. Using Wikipedia for automatic Word Sense Disambiguation. In *Proc. of NAACL-HLT-07*, pages 196–203.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, N.J.

David Milne and Ian H. Witten. 2008a. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy at AAAI-08*, pages 25–30.

David Milne and Ian H. Witten. 2008b. Learning to link with Wikipedia. In *Proc. of CIKM-08*, pages 509–518.

Vivi Nastase and Michael Strube. 2008. Decoding Wikipedia category names for knowledge acquisition. In *Proc. of AAAI-08*, pages 1219–1224.

Vivi Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and activation spreading. In *Proc. of EMNLP-08*, pages 763–772.

Roberto Navigli and Mirella Lapata. 2010. An experimental study on graph connectivity for unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Anaylsis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proc. of ACL-10*.

Roberto Navigli and Paola Velardi. 2005. Structural Semantic Interconnections: a knowledge-based approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1075–1088.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained English all-words task. In *Proc. of SemEval-2007*, pages 30–35.

Roberto Navigli. 2009a. Using cycles and quasi-cycles to disambiguate dictionary glosses. In *Proc. of EACL-09*, pages 594–602.

Roberto Navigli. 2009b. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Marco Pennacchiotti and Patrick Pantel. 2006. Ontologizing semantic relations. In *Proc. of COLING-ACL-06*, pages 793–800.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proc. of IJCAI-09*, pages 2083–2088.

Simone Paolo Ponzetto and Michael Strube. 2007a. Deriving a large scale taxonomy from Wikipedia. In *Proc. of AAAI-07*, pages 1440–1445.

Simone Paolo Ponzetto and Michael Strube. 2007b. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Nils Reiter, Matthias Hartung, and Anette Frank. 2008. A resource-poor approach for linking ontology classes to Wikipedia articles. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing*, volume 1 of *Research in Computational Semantics*, pages 381–387. College Publications, London, England.

German Rigau, Horacio Rodríguez, and Eneko Agirre. 1998. Building accurate semantic taxonomies from monolingual MRDs. In *Proc. of COLING-ACL-98*, pages 1103–1109.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic assignment of Wikipedia encyclopedic entries to WordNet synsets. In *Advances in Web Intelligence*, volume 3528 of *Lecture Notes in Computer Science*. Springer Verlag.

Christina Sauper and Regina Barzilay. 2009. Automatically generating Wikipedia articles: A structure-aware approach. In *Proc. of ACL-IJCNLP-09*, pages 208–216.

Eyal Shnarch, Libby Barak, and Ido Dagan. 2009. Extracting lexical reference rules from Wikipedia. In *Proc. of ACL-IJCNLP-09*, pages 450–458.

Rion Snow, Dan Jurafsky, and Andrew Ng. 2006. Semantic taxonomy induction from heterogeneous evidence. In *Proc. of COLING-ACL-06*, pages 801–808.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. Yago: A large ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6(3):203–217.

Fei Wu and Daniel Weld. 2007. Automatically semantifying Wikipedia. In *Proc. of CIKM-07*, pages 41–50.

Fei Wu and Daniel Weld. 2008. Automatically refining the Wikipedia infobox ontology. In *Proc. of WWW-08*, pages 635–644.

All Words Domain Adapted WSD: Finding a Middle Ground between Supervision and Unsupervision

Mitesh M. Khapra Anup Kulkarni Saurabh Sohoney Pushpak Bhattacharyya
Indian Institute of Technology Bombay,
Mumbai - 400076, India.
{miteshk,anup,saurabhsohoney,pb}@cse.iitb.ac.in

Abstract

In spite of decades of research on word sense disambiguation (WSD), all-words general purpose WSD has remained a distant goal. Many supervised WSD systems have been built, but the effort of creating the training corpus - *annotated sense marked corpora* - has always been a matter of concern. Therefore, attempts have been made to develop unsupervised and knowledge based techniques for WSD which do not need sense marked corpora. However such approaches have not proved effective, since they typically do not better Wordnet first sense baseline accuracy. Our research reported here proposes to stick to the supervised approach, but with far less demand on annotation. We show that if we have ANY sense marked corpora, be it from mixed domain or a specific domain, a small amount of annotation in ANY other domain can deliver the goods almost as if exhaustive sense marking were available in that domain. We have tested our approach across Tourism and Health domain corpora, using also the well known mixed domain SemCor corpus. Accuracy figures close to self domain training lend credence to the viability of our approach. Our contribution thus lies in finding a convenient middle ground between pure supervised and pure unsupervised WSD. Finally, our approach is not restricted to any specific set of target words, a departure from a commonly observed practice in domain specific WSD.

1 Introduction

Amongst annotation tasks, sense marking surely takes the cake, demanding as it does high level of language competence, topic comprehension and domain sensitivity. This makes supervised approaches to WSD a difficult proposition (Agirre et al., 2009b; Agirre et al., 2009a; McCarthy et al., 2007). Unsupervised and knowledge based approaches have been tried with the hope of creating WSD systems with no need for sense marked corpora (Koeling et al., 2005; McCarthy et al., 2007; Agirre et al., 2009b). However, the accuracy figures of such systems are low.

Our work here is motivated by the desire to develop *annotation-lean all-words* domain adapted techniques for supervised WSD. It is a common observation that domain specific WSD exhibits high level of accuracy even for the all-words scenario (Khapra et al., 2010) - provided training and testing are on the same domain. Also domain adaptation - in which training happens in one domain and testing in another - often is able to attain good levels of performance, albeit on a specific set of target words (Chan and Ng, 2007; Agirre and de Lacalle, 2009). To the best of our knowledge there does not exist a system that solves the combined problem of *all words domain adapted WSD*. We thus propose the following:

a. For any target domain, create a small amount of sense annotated corpus.

b. Mix it with an existing sense annotated corpus – from a mixed domain or specific domain – to train the WSD engine.

This procedure tested on four adaptation scenarios, *viz.*, (i) SemCor (Miller et al., 1993) to Tourism, (ii) SemCor to Health, (iii) Tourism to Health and (iv) Health to Tourism has consistently yielded good performance (to be explained in sections 6 and 7).

The remainder of this paper is organized as follows. In section 2 we discuss previous work in the area of domain adaptation for WSD. In section 3

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1532–1541,
Uppsala, Sweden, 11-16 July 2010.

we discuss three state of art supervised, unsupervised and knowledge based algorithms for WSD. Section 4 discusses the injection strategy for domain adaptation. In section 5 we describe the dataset used for our experiments. We then present the results in section 6 followed by discussions in section 7. Section 8 examines whether there is any need for intelligent choice of injections. Section 9 concludes the paper highlighting possible future directions.

2 Related Work

Domain specific WSD for selected target words has been attempted by Ng and Lee (1996), Agirre and de Lacalle (2009), Chan and Ng (2007), Koeling et al. (2005) and Agirre et al. (2009b). They report results on three publicly available lexical sample datasets, *viz.*, DSO corpus (Ng and Lee, 1996), MEDLINE corpus (Weeber et al., 2001) and the corpus made available by Koeling et al. (2005). Each of these datasets contains a handful of target words (41-191 words) which are sense marked in the corpus.

Our main inspiration comes from the target-word specific results reported by Chan and Ng (2007) and Agirre and de Lacalle (2009). The former showed that adding just 30% of the target data to the source data achieved the same performance as that obtained by taking the entire source and target data. Agirre and de Lacalle (2009) reported a 22% error reduction when source and target data were combined for training a classifier, as compared to the case when only the target data was used for training the classifier. However, both these works focused on *target word specific* WSD and do not address all-words domain specific WSD.

In the unsupervised setting, McCarthy et al. (2007) showed that their predominant sense acquisition method gives good results on the corpus of Koeling et al. (2005). In particular, they showed that the performance of their method is comparable to the most frequent sense obtained from a tagged corpus, thereby making a strong case for unsupervised methods for domain-specific WSD. More recently, Agirre et al. (2009b) showed that knowledge based approaches which rely only on the semantic relations captured by the Wordnet graph outperform supervised approaches when applied to specific domains. The good results obtained by McCarthy et al. (2007) and Agirre et al. (2009b) for unsupervised and knowledge based approaches respectively have cast a doubt on the viability of supervised approaches which rely on sense tagged corpora. However, these conclusions were drawn only from the performance on certain target words, leaving open the question of their utility in all words WSD.

We believe our work contributes to the WSD research in the following way: (i) it shows that there is promise in supervised approach to all-word WSD, through the instrument of domain adaptation; (ii) it places in perspective some very recently reported unsupervised and knowledge based techniques of WSD; (ii) it answers some questions arising out of the debate between supervision and unsupervision in WSD; and finally (iv) it explores a convenient middle ground between unsupervised and supervised WSD – the territory of "annotate-little and inject" paradigm.

3 WSD algorithms employed by us

In this section we describe the knowledge based, unsupervised and supervised approaches used for our experiments.

3.1 Knowledge Based Approach

Agirre et al. (2009b) showed that a graph based algorithm which uses only the relations between concepts in a Lexical Knowledge Base (LKB) can outperform supervised approaches when tested on specific domains (for a set of chosen target words). We employ their method which involves the following steps:

1. Represent Wordnet as a graph where the concepts (*i.e.*, synsets) act as nodes and the relations between concepts define edges in the graph.
2. Apply a context-dependent *Personalized PageRank* algorithm on this graph by introducing the context words as nodes into the graph and linking them with their respective synsets.
3. These nodes corresponding to the context words then inject probability mass into the synsets they are linked to, thereby influencing the final relevance of all nodes in the graph.

We used the publicly available implementation of this algorithm[1] for our experiments.

[1] http://ixa2.si.ehu.es/ukb/

3.2 Unsupervised Approach

McCarthy et al. (2007) used an untagged corpus to construct a thesaurus of related words. They then found the predominant sense (i.e., the most frequent sense) of each target word using pair-wise Wordnet based similarity measures by pairing the target word with its top-k neighbors in the thesaurus. Each target word is then disambiguated by assigning it its predominant sense – the motivation being that the predominant sense is a powerful, hard-to-beat baseline. We implemented their method using the following steps:

1. Obtain a domain-specific untagged corpus (we crawled a corpus of approximately 9M words from the web).
2. Extract grammatical relations from this text using a dependency parser[2] (Klein and Manning, 2003).
3. Use the grammatical relations thus extracted to construct features for identifying the k nearest neighbors for each word using the distributional similarity score described in (Lin, 1998).
4. Rank the senses of each target word in the test set using a weighted sum of the distributional similarity scores of the neighbors. The weights in the sum are based on Wordnet Similarity scores (Patwardhan and Pedersen, 2003).
5. Each target word in the test set is then disambiguated by simply assigning it its predominant sense obtained using the above method.

3.3 Supervised approach

Khapra et al. (2010) proposed a supervised algorithm for domain-specific WSD and showed that it beats the most frequent corpus sense and performs on par with other state of the art algorithms like PageRank. We implemented their iterative algorithm which involves the following steps:

1. Tag all monosemous words in the sentence.
2. Iteratively disambiguate the remaining words in the sentence in increasing order of their degree of polysemy.
3. At each stage rank the candidate senses of a word using the scoring function of Equation (1) which combines corpus based parameters (such as, sense distributions and corpus co-occurrence) and Wordnet based parameters (such as, semantic similarity, conceptual distance, *etc.*)

$$S^* = \arg\max_i(\theta_i V_i + \sum_{j \in J} W_{ij} * V_i * V_j) \quad (1)$$

where,

$i \in$ *Candidate Synsets*
$J =$ *Set of disambiguated words*
$\theta_i = BelongingnessToDominantConcept(S_i)$
$V_i = P(S_i|word)$
$W_{ij} = CorpusCooccurrence(S_i, S_j)$
$* 1/WNConceptualDistance(S_i, S_j)$
$* 1/WNSemanticGraphDistance(S_i, S_j)$

4. Select the candidate synset with maximizes the above score as the winner sense.

4 Injections for Supervised Adaptation

This section describes the main interest of our work i.e. *adaptation using injections*. For supervised adaptation, we use the supervised algorithm described above (Khapra et al., 2010) in the following 3 settings as proposed by Agirre et al. (2009a):

a. **Source setting:** We train the algorithm on a mixed-domain corpus (SemCor) or a domain-specific corpus (say, Tourism) and test it on a different domain (say, Health). A good performance in this setting would indicate robustness to domain-shifts.

b. **Target setting:** We train and test the algorithm using data from the same domain. This gives the skyline performance, i.e., the best performance that can be achieved if sense marked data from the target domain were available.

c. **Adaptation setting:** This setting is the main focus of interest in the paper. We augment the training data which could be from one domain or mixed domain with a small amount of data from the target domain. This combined data is then used for training. The aim here is to reach as close to the skyline performance using as little data as possible. For injecting data from the target domain we randomly select some sense marked words from the target domain and add

[2]We used the Stanford parser - http://nlp.stanford.edu/software/lex-parser.shtml

Category	Polysemous words		Monosemous words	
	Tourism	Health	Tourism	Health
Noun	53133	15437	23665	6979
Verb	15528	7348	1027	356
Adjective	19732	5877	10569	2378
Adverb	6091	1977	4323	1694
All	94484	30639	39611	11407

Table 1: Polysemous and Monosemous words per category in each domain

Category	Avg. no. of instances per polysemous word		
	Health	Tourism	SemCor
Noun	7.06	12.56	10.98
Verb	7.47	9.76	11.95
Adjective	5.74	12.07	8.67
Adverb	9.11	19.78	25.44
All	6.94	12.17	11.25

Table 2: Average number of instances per polysemous word per category in the 3 domains

Category	Avg. degree of Wordnet polysemy for polysemous words		
	Health	Tourism	SemCor
Noun	5.24	4.95	5.60
Verb	10.60	10.10	9.89
Adjective	5.52	5.08	5.40
Adverb	3.64	4.16	3.90
All	6.49	5.77	6.43

Table 3: Average degree of Wordnet polysemy of polysemous words per category in the 3 domains

Category	Avg. degree of Corpus polysemy for polysemous words		
	Health	Tourism	SemCor
Noun	1.92	2.60	3.41
Verb	3.41	4.55	4.73
Adjective	2.04	2.57	2.65
Adverb	2.16	2.82	3.09
All	2.31	2.93	3.56

Table 4: Average degree of Corpus polysemy of polysemous words per category in the 3 domains

them to the training data. An obvious question which arises at this point is "Why were the words selected at random?" or "Can selection of words using some active learning strategy yield better results than a random selection?" We discuss this question in detail in Section 7 and show that a random set of injections performs no worse than a craftily selected set of injections.

5 DataSet Preparation

Due to the lack of any publicly available all-words domain specific sense marked corpora we set upon the task of collecting data from two domains, *viz.*, *Tourism and Health*. The data for Tourism domain was downloaded from Indian Tourism websites whereas the data for Health domain was obtained from two doctors. This data was manually sense annotated by two lexicographers adept in English. Princeton Wordnet 2.1[3] (Fellbaum, 1998) was used as the sense inventory. A total of 1,34,095 words from the Tourism domain and 42,046 words from the Health domain were manually sense marked. Some files were sense marked by both the lexicographers and the Inter Tagger Agreement (ITA) calculated from these files was 83% which is comparable to the 78% ITA reported on the SemCor corpus considering the domain-specific nature of the corpus.

We now present different statistics about the corpora. Table 1 summarizes the number of polysemous and monosemous words in each category. Note that we do not use the monosemous words while calculating precision and recall of our algorithms.

Table 2 shows the average number of instances per polysemous word in the 3 corpora. We note that the number of instances per word in the Tourism domain is comparable to that in the SemCor corpus whereas the number of instances per word in the Health corpus is smaller due to the overall smaller size of the Health corpus.

Tables 3 and 4 summarize the average degree of Wordnet polysemy and corpus polysemy of the polysemous words in the corpus. Wordnet polysemy is the number of senses of a word as listed in the Wordnet, whereas corpus polysemy is the number of senses of a word actually appearing in the corpus. As expected, the average degree of corpus polysemy (Table 4) is much less than the average degree of Wordnet polysemy (Table 3). Further, the average degree of corpus polysemy (Table 4) in the two domains is less than that in the mixed-domain SemCor corpus, which is expected due to the domain specific nature of the corpora.

Finally, Table 5 summarizes the number of unique polysemous words per category in each domain.

Category	No. of unique polysemous words		
	Health	Tourism	SemCor
Noun	2188	4229	5871
Verb	984	1591	2565
Adjective	1024	1635	2640
Adverb	217	308	463
All	4413	7763	11539

Table 5: Number of unique polysemous words per category in each domain.

[3] http://wordnetweb.princeton.edu/perl/webwn

The data is currently being enhanced by manually sense marking more words from each domain and will be soon freely available[4] for research purposes.

6 Results

We tested the 3 algorithms described in section 4 using SemCor, Tourism and Health domain corpora. We did a 2-fold cross validation for supervised adaptation and report the average performance over the two folds. Since the knowledge based and unsupervised methods do not need any training data we simply test it on the entire corpus from the two domains.

6.1 Knowledge Based approach

The results obtained by applying the Personalized PageRank (PPR) method to Tourism and Health data are summarized in Table 6. We also report the Wordnet first sense baseline (WFS).

Domain	Algorithm	P(%)	R(%)	F(%)
Tourism	PPR	53.1	53.1	53.1
	WFS	62.5	62.5	62.5
Health	PPR	51.1	51.1	51.1
	WFS	65.5	65.5	65.5

Table 6: Comparing the performance of Personalized PageRank (PPR) with Wordnet First Sense Baseline (WFS)

6.2 Unsupervised approach

The predominant sense for each word in the two domains was calculated using the method described in section 4.2. McCarthy et al. (2004) reported that the best results were obtained using $k = 50$ neighbors and the Wordnet Similarity ***jcn*** measure (Jiang and Conrath, 1997). Following them, we used $k = 50$ and observed that the best results for nouns and verbs were obtained using the ***jcn*** measure and the best results for adjectives and adverbs were obtained using the ***lesk*** measure (Banerjee and Pedersen, 2002). Accordingly, we used ***jcn*** for nouns and verbs and ***lesk*** for adjectives and adverbs. Each target word in the test set is then disambiguated by simply assigning it its predominant sense obtained using the above method. We tested this approach only on Tourism domain due to unavailability of large untagged Health corpus which is needed for constructing the thesaurus. The results are summarized in Table 7.

Domain	Algorithm	P(%)	R(%)	F(%)
Tourism	McCarthy	51.85	49.32	50.55
	WFS	62.50	62.50	62.50

Table 7: Comparing the performance of unsupervised approach with Wordnet First Sense Baseline (WFS)

6.3 Supervised adaptation

We report results in the **source setting**, **target setting** and **adaptation setting** as described earlier using the following four combinations for source and target data:

1. ***SemCor to Tourism (SC→T)*** where SemCor is used as the source domain and Tourism as the target (test) domain.
2. ***SemCor to Health (SC→H)*** where SemCor is used as the source domain and Health as the target (test) domain.
3. ***Tourism to Health (T→H)*** where Tourism is used as the source domain and Health as the target (test) domain.
4. ***Health to Tourism (H→T)*** where Health is used as the source domain and Tourism as the target (test) domain.

In each case, the target domain data was divided into two folds. One fold was set aside for testing and the other for injecting data in the ***adaptation setting***. We increased the size of the injected target examples from 1000 to 14000 words in increments of 1000. We then repeated the same experiment by reversing the role of the two folds.

Figures 1, 2, 3 and 4 show the graphs of the average F-score over the 2-folds for SC→T, SC→H, T→H and H→T respectively. The x-axis represents the amount of training data (in words) injected from the target domain and the y-axis represents the F-score. The different curves in each graph are as follows:

a. *only_random* : This curve plots the performance obtained using x randomly selected sense tagged words from the target domain and zero sense tagged words from the source domain (x was varied from 1000 to 14000 words in increments of 1000).

[4] http://www.cfilt.iitb.ac.in/wsd/annotated_corpus

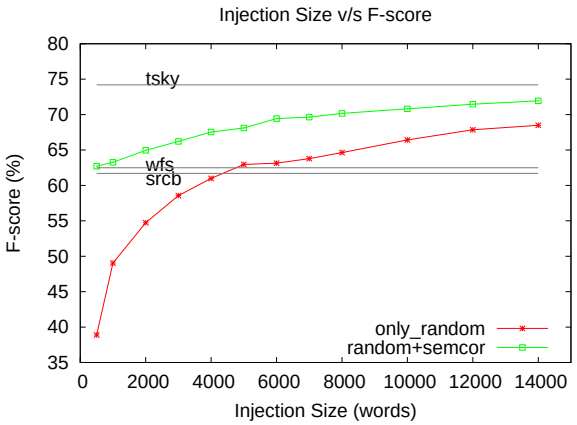


Figure 1: Supervised adaptation from SemCor to Tourism using injections

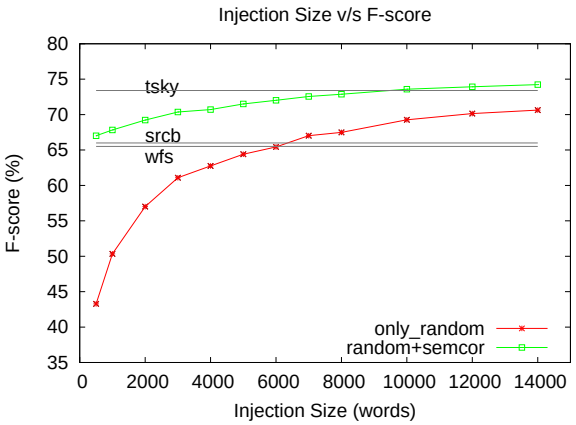


Figure 2: Supervised adaptation from SemCor to Health using injections

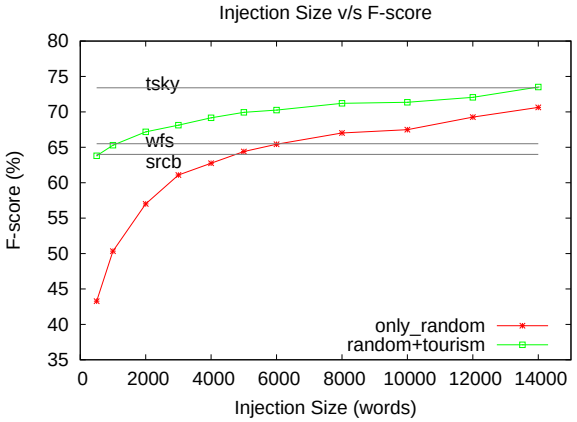


Figure 3: Supervised adaptation from Tourism to Health using injections

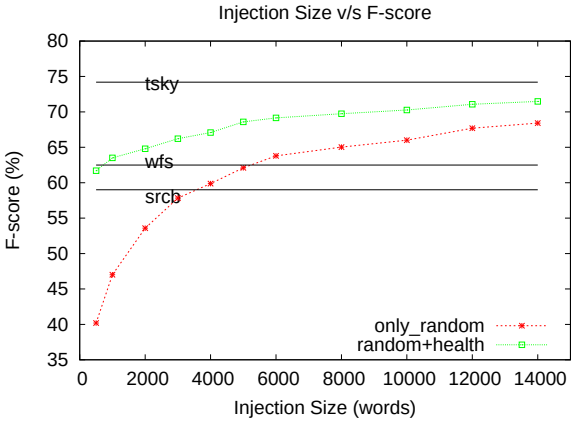


Figure 4: Supervised adaptation from Health to Tourism using injections

b. *random+source* : This curve plots the performance obtained by mixing x randomly selected sense tagged words from the target domain with the entire training data from the source domain (again x was varied from 1000 to 14000 words in increments of 1000).

c. *source_baseline* ***(srcb)*** : This represents the F-score obtained by training on the source data alone without mixing any examples from the target domain.

d. *wordnet_first_sense* ***(wfs)*** : This represents the F-score obtained by selecting the first sense from Wordnet, a typically reported baseline.

e. *target_skyline* ***(tsky)*** : This represents the average 2-fold F-score obtained by training on one entire fold of the target data itself (*Health*: 15320 polysemous words; *Tourism*: 47242 polysemous words) and testing on the other fold.

These graphs along with other results are discussed in the next section.

7 Discussions

We discuss the performance of the three approaches.

7.1 Knowledge Based and Unsupervised approaches

It is apparent from Tables 6 and 7 that knowledge based and unsupervised approaches do not perform well when compared to the Wordnet first sense (which is freely available and hence can be used for disambiguation). Further, we observe that the performance of these approaches is even less than the *source_baseline* (*i.e.*, the case when training data from a source domain is applied as it is to a target domain - without using any injections). These observations bring out the weaknesses of these approaches when used in an all-words setting and clearly indicate that they come nowhere close to replacing a supervised system.

7.2 Supervised adaptation

1. The F-score obtained by training on SemCor (mixed-domain corpus) and testing on the two target domains without using any injections (***srcb***) – F-score of 61.7% on Tourism and F-score of 65.5% on Health – is comparable to the best result reported on the SEMEVAL datasets (65.02%, where both training and testing happens on a mixed-domain corpus (Snyder and Palmer, 2004)). This is in contrast to previous studies (Escudero et al., 2000; Agirre and Martinez, 2004) which suggest that instead of adapting from a generic/mixed domain to a specific domain, it is better to completely ignore the generic examples and use hand-tagged data from the target domain itself. The main reason for the contrasting results is that the earlier work focused only on a handful of target words whereas we focus on all words appearing in the corpus. So, while the behavior of a few target words would change drastically when the domain changes, a majority of the words will exhibit the same behavior (*i.e.*, same predominant sense) even when the domain changes. We agree that the overall performance is still lower than that obtained by training on the domain-specific corpora. However, it is still better than the performance of unsupervised and knowledge based approaches which tilts the scale in favor of supervised approaches even when only mixed domain sense marked corpora is available.

2. Adding injections from the target domain improves the performance. As the amount of injection increases the performance approaches the skyline, and in the case of SC→H and T→H it even crosses the skyline performance showing that combining the source and target data can give better performance than using the target data alone. This is consistent with the domain adaptation results reported by Agirre and de Lacalle (2009) on a specific set of target words.

3. The performance of *random+source* is always better than *only_random* indicating that the data from the source domain does help to improve performance. A detailed analysis showed that the gain obtained by using the source data is attributable to reducing recall errors by increasing the coverage of seen words.

4. Adapting from one specific domain (*Tourism or Health*) to another specific domain (*Health or Tourism*) gives the same performance as that obtained by adapting from a mixed-domain (*SemCor*) to a specific domain (*Tourism, Health*). This is an interesting observation as it suggests that as long as data from one domain is available it is easy to build a WSD engine that works for other domains by injecting a small amount of data from these domains.

To verify that the results are consistent, we randomly selected 5 different sets of injections from fold-1 and tested the performance on fold-2. We then repeated the same experiment by reversing the roles of the two folds. The results were indeed consistent irrespective of the set of injections used. Due to lack of space we have not included the results for these 5 different sets of injections.

7.3 Quantifying the trade-off between performance and corpus size

To correctly quantify the benefit of adding injections from the target domain, we calculated the amount of target data (*peak_size*) that is needed to reach the skyline F-score (*peak_F*) in the absence of any data from the source domain. The *peak_size* was found to be 35000 (Tourism) and 14000 (Health) corresponding to *peak_F* values of 74.2% (Tourism) and 73.4% (Health). We then plotted a graph (Figure 5) to capture the relation between the size of injections (expressed as a percentage of the *peak_size*) and the F-score (expressed as a percentage of the *peak_F*).

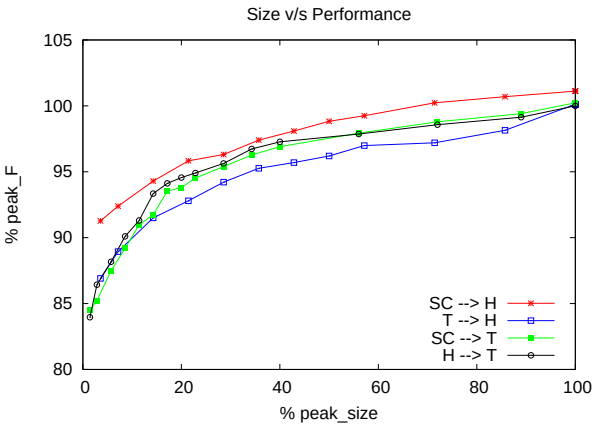


Figure 5: Trade-off between performance and corpus size

We observe that by mixing only 20-40% of the *peak_size* with the source domain we can obtain up to 95% of the performance obtained by using the

entire target data (*peak_size*). In absolute terms, the size of the injections is only 7000-9000 polysemous words which is a very small price to pay considering the performance benefits.

8 Does the choice of injections matter?

An obvious question which arises at this point is *"Why were the words selected at random?"* or *"Can selection of words using some active learning strategy yield better results than a random selection?"* An answer to this question requires a more thorough understanding of the *sense-behavior* exhibited by words across domains. In any scenario involving a shift from domain D_1 to domain D_2, we will always encounter words belonging to the following 4 categories:

a. W_{D_1} : This class includes words which are encountered only in the source domain D_1 and do not appear in the target domain D_2. Since we are interested in adapting to the target domain and since these words do not appear in the target domain, it is quite obvious that they are **not important** for the problem of domain adaptation.

b. W_{D_2} : This class includes words which are encountered only in the target domain D_2 and do not appear in the source domain D_1. Again, it is quite obvious that these words are **important** for the problem of domain adaptation. They fall in the category of unseen words and need handling from that point of view.

c. $W_{D1D2_{conformists}}$: This class includes words which are encountered in both the domains and exhibit the same predominant sense in both the domains. Correct identification of these words is **important** so that we can use the predominant sense learned from D_1 for disambiguating instances of these words appearing in D_2.

d. $W_{D1D2_{non-conformists}}$: This class includes words which are encountered in both the domains but their predominant sense in the target domain D_2 **does not conform** to the predominant sense learned from the source domain D_1. Correct identification of these words is **important** so that we can ignore the predominant senses learned from D_1 while disambiguating instances of these words appearing in D_2.

Table 8 summarizes the percentage of words that fall in each category in each of the three adaptation scenarios. The fact that nearly 50-60% of the words fall in the "conformist" category once again makes a strong case for reusing sense tagged data from one domain to another domain.

Category	SC→T	SC→H	T→H
$\mathbf{W_{D_2}}$	7.14%	5.45%	13.61%
Conformists	49.54%	60.43%	54.31%
Non-Conformists	43.30%	34.11%	32.06%

Table 8: Percentage of Words belonging to each category in the three settings.

The above characterization suggests that an *ideal* domain adaptation strategy should focus on injecting W_{D_2} and $W_{D1D2_{non-conformists}}$ as these would yield maximum benefits if injected into the training data. While it is easy to identify the W_{D_2} words, *"identifying non-conformists"* is a hard problem which itself requires some type of WSD[5]. However, just to prove that a *random* injection strategy does as good as an *ideal* strategy we assume the presence of an *oracle* which identifies the $W_{D1D2_{non-conformists}}$. We then augment the training data with 5-8 instances for W_{D_2} and $W_{D1D2_{non-conformists}}$ words thus identified. We observed that adding more than 5-8 instances per word does not improve the performance. This is due to the "one sense per domain" phenomenon – seeing only a few instances of a word is sufficient to identify the predominant sense of the word. Further, to ensure a better overall performance, the instances of the most frequent words are injected first followed by less frequent words till we exhaust the total size of the injections (1000, 2000 and so on). We observed that there was a 75-80% overlap between the words selected by random strategy and oracle strategy. This is because oracle selects the most frequent words which also have a high chance of getting selected when a random sampling is done.

Figures 6, 7, 8 and 9 compare the performance of the two strategies. We see that the random strategy does as well as the oracle strategy thereby supporting our claim that *if we have sense marked corpus from one domain then simply injecting ANY small amount of data from the target domain will*

[5]Note that the unsupervised predominant sense acquisition method of McCarthy et al. (2007) implicitly identifies conformists and non-conformists

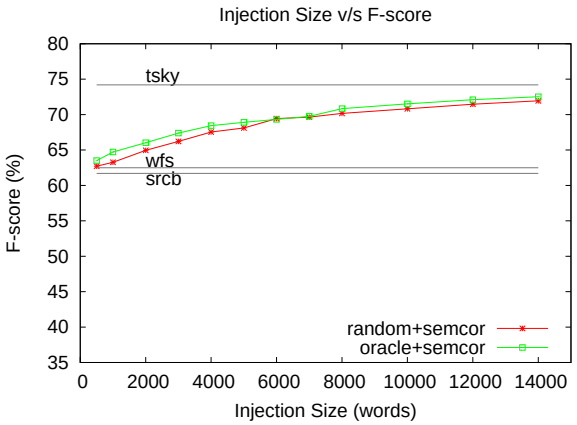


Figure 6: Comparing random strategy with oracle based ideal strategy for SemCor to Tourism adaptation

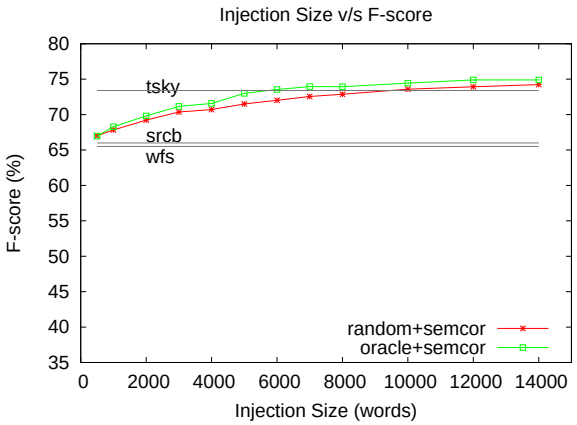


Figure 7: Comparing random strategy with oracle based ideal strategy for SemCor to Health adaptation

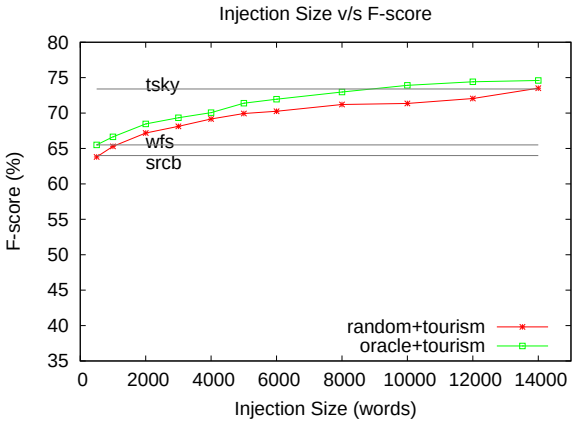


Figure 8: Comparing random strategy with oracle based ideal strategy for Tourism to Health adaptation

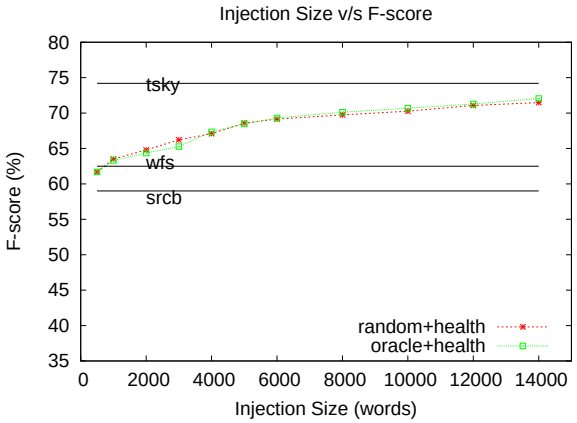


Figure 9: Comparing random strategy with oracle based ideal strategy for Health to Tourism adaptation

do the job.

9 Conclusion and Future Work

Based on our study of WSD in 4 domain adaptation scenarios, we make the following conclusions:

1. Supervised adaptation by mixing small amount of data (7000-9000 words) from the target domain with the source domain gives nearly the same performance (F-score of around 70% in all the 4 adaptation scenarios) as that obtained by training on the entire target domain data.

2. Unsupervised and knowledge based approaches which use distributional similarity and Wordnet based similarity measures do not compare well with the Wordnet first sense baseline performance and do not come anywhere close to the performance of supervised adaptation.

3. Supervised adaptation from a mixed domain to a specific domain gives the same performance as that from one specific domain (Tourism) to another specific domain (Health).

4. Supervised adaptation is not sensitive to the type of data being injected. This is an interesting finding with the following implication: as long as one has sense marked corpus - be it from a mixed or specific domain - simply injecting ANY small amount of data from the target domain suffices to beget good accuracy.

As future work, we would like to test our work on the Environment domain data which was released as part of the SEMEVAL 2010 shared task on "All-words Word Sense Disambiguation on a Specific Domain".

References

Eneko Agirre and Oier Lopez de Lacalle. 2009. Supervised domain adaption for wsd. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 42–50, Morristown, NJ, USA. Association for Computational Linguistics.

Eneko Agirre and David Martinez. 2004. The effect of bias on an automatically-built word sense corpus. In *Proceedings of the 4rd International Conference on Languages Resources and Evaluations (LREC)*.

Eneko Agirre, Oier Lopez de Lacalle, Christiane Fellbaum, Andrea Marchetti, Antonio Toral, and Piek Vossen. 2009a. Semeval-2010 task 17: all-words word sense disambiguation on a specific domain. In *DEW '09: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 123–128, Morristown, NJ, USA. Association for Computational Linguistics.

Eneko Agirre, Oier Lopez De Lacalle, and Aitor Soroa. 2009b. Knowledge-based wsd on specific domains: Performing better than generic supervised wsd. In *In Proceedings of IJCAI*.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 136–145, London, UK. Springer-Verlag.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 49–56, Prague, Czech Republic, June. Association for Computational Linguistics.

Gerard Escudero, Lluís Màrquez, and German Rigau. 2000. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 172–180, Morristown, NJ, USA. Association for Computational Linguistics.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*.

J.J. Jiang and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33.

Mitesh Khapra, Sapan Shah, Piyush Kedia, and Pushpak Bhattacharyya. 2010. Domain-specific word sense disambiguation combining corpus based and wordnet based parameters. In *5th International Conference on Global Wordnet (GWC2010)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *IN PROCEEDINGS OF THE 41ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, pages 423–430.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 419–426, Morristown, NJ, USA. Association for Computational Linguistics.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279, Morristown, NJ, USA. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2007. Unsupervised acquisition of predominant word senses. *Comput. Linguist.*, 33(4):553–590.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 303–308, Morristown, NJ, USA. Association for Computational Linguistics.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 40–47, Morristown, NJ, USA. Association for Computational Linguistics.

Siddharth Patwardhan and Ted Pedersen. 2003. The cpan wordnet::similarity package. http://search .cpan.org/ sid/wordnet-similarity/.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July. Association for Computational Linguistics.

Marc Weeber, James G. Mork, and Alan R. Aronson. 2001. Developing a test collection for biomedical word sense disambiguation. In *In Proceedings of the AMAI Symposium*, pages 746–750.

Combining Orthogonal Monolingual and Multilingual Sources of Evidence for All Words WSD

Weiwei Guo
Computer Science Department
Columbia University
New York, NY, 10115
weiwei@cs.columbia.edu

Mona Diab
Center for Computational Learning Systems
Columbia University
New York, NY, 10115
mdiab@ccls.columbia.edu

Abstract

Word Sense Disambiguation remains one of the most complex problems facing computational linguists to date. In this paper we present a system that combines evidence from a monolingual WSD system together with that from a multilingual WSD system to yield state of the art performance on standard All-Words data sets. The monolingual system is based on a modification of the graph based state of the art algorithm In-Degree. The multilingual system is an improvement over an All-Words unsupervised approach, SALAAM. SALAAM exploits multilingual evidence as a means of disambiguation. In this paper, we present modifications to both of the original approaches and then their combination. We finally report the highest results obtained to date on the SENSEVAL 2 standard data set using an unsupervised method, we achieve an overall F measure of 64.58 using a voting scheme.

1 Introduction

Despite advances in natural language processing (NLP), Word Sense Disambiguation (WSD) is still considered one of the most challenging problems in the field. Ever since the field's inception, WSD has been perceived as one of the central problems in NLP. WSD is viewed as an enabling technology that could potentially have far reaching impact on NLP applications in general. We are starting to see the beginnings of a positive effect of WSD in NLP applications such as Machine Translation (Carpuat and Wu, 2007; Chan et al., 2007).

Advances in WSD research in the current millennium can be attributed to several key factors: the availability of large scale computational lexical resources such as WordNets (Fellbaum, 1998; Miller, 1990), the availability of large scale corpora, the existence and dissemination of standardized data sets over the past 10 years through different testbeds such as SENSEVAL and SEMEVAL competitions,[1] devising more robust computing algorithms to handle large scale data sets, and simply advancement in hardware machinery.

In this paper, we address the problem of WSD of all content words in a sentence, All-Words data. In this framework, the task is to associate all tokens with their contextually relevant meaning definitions from some computational lexical resource. Our work hinges upon combining two high quality WSD systems that rely on essentially different sources of evidence. The two WSD systems are a monolingual system `RelCont` and a multilingual system `TransCont`. `RelCont` is an enhancement on an existing graph based algorithm, In-Degree, first described in (Navigli and Lapata, 2007). `TransCont` is an enhancement over an existing approach that leverages multilingual evidence through projection, `SALAAM`, described in detail in (Diab and Resnik, 2002). Similar to the leveraged systems, the current combined approach is unsupervised, namely it does not rely on training data from the onset. We show that by combining both sources of evidence, our approach yields the highest performance for an unsupervised system to date on standard All-Words data sets.

This paper is organized as follows: Section 2 delves into the problem of WSD in more detail; Section 3 explores some of the relevant related work; in Section 4, we describe the two WSD systems in some detail emphasizing the improvements to the basic systems in addition to a description of our combination approach; we present our experimental set up and results in Section 5; we discuss the results and our overall observations with error analysis in Section 6; Finally, we con-

[1] http://www.semeval.org

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1542–1551,
Uppsala, Sweden, 11-16 July 2010.

clude in Section 7.

2 Word Sense Disambiguation

The definition of WSD has taken on several different practical meanings in recent years. In the latest SEMEVAL 2010 workshop, there are 18 tasks defined, several of which are on different languages, however we recognize the widening of the definition of the task of WSD. In addition to the traditional All-Words and Lexical Sample tasks, we note new tasks on word sense discrimination (no sense inventory needed, the different senses are merely distinguished), lexical substitution using synonyms of words as substitutes both monolingually and multilingually, as well as meaning definitions obtained from different languages namely using words in translation.

Our paper is about the classical All-Words (AW) task of WSD. In this task, all content bearing words in running text are disambiguated from a static lexical resource. For example a sentence such as 'I walked by the bank and saw many beautiful plants there.' will have the verbs 'walked, saw', the nouns 'bank, plants', the adjectives 'many, beautiful', and the adverb 'there', be disambiguated from a standard lexical resource. Hence, using WordNet,[2] 'walked' will be assigned the corresponding meaning definitions of: *t*o use one's feet to advance; to advance by steps, 'saw' will be assigned the meaning definition of: *t*o perceive by sight or have the power to perceive by sight, the noun 'bank' will be assigned the meaning definition of: *s*loping land especially the slope beside a body of water, and so on.

3 Related Works

Many systems over the years have been proposed for the task. A thorough review of the state of the art through the late 1990s (Ide and Veronis, 1998) and more recently in (Navigli, 2009). Several techniques have been used to tackle the problem ranging from rule based/knowledge based approaches to unsupervised and supervised machine learning techniques. To date, the best approaches that solve the AW WSD task are supervised as illustrated in the different SenseEval and SEMEVAL AW task (Palmer et al., 2001; Snyder and Palmer, 2004; Pradhan et al., 2007).

In this paper, we present an unsupervised combination approach to the AW WSD problem that relies on WN similarity measures in conjunction with evidence obtained through exploiting multilingual evidence. We will review the closely relevant related work on which this current investigation is based.[3]

4 Our Approach

Our current investigation exploits two basic unsupervised approaches that perform at state-of-the-art for the AW WSD task in an unsupervised setting. Crucially the two systems rely on different sources of evidence allowing them to complement each other to a large extent leading to better performance than for each system independently. Given a target content word and co-occurring contextual clues, the monolingual system `RelCont` attempts to assign the approporiate meaning definition to the target word. Such words by definition are semantically related words. `TransCont`, on the other hand, is the multilingual system. `TransCont` defines the notion of context in the translational space using a foreign word as a filter for defining the contextual content words for a given target word. In this multilingual setting, all the words that are mapped to (aligned with) the same orthographic form in a foreign language constitute the context. In the next subsections we describe the two approaches `RelCont` and `TransCont` in some detail, then we proceed to describe two combination methods for the two approaches: `MERGE` and `VOTE`.

4.1 Monolingual System `RelCont`

`RelCont` is based on an extension of a state-of-the-art WSD approach by (Sinha and Mihalcea, 2007), henceforth (SM07). In the basic SM07 work, the authors combine different semantic similarity measures with different graph based algorithms as an extension to work in (Mihalcea, 2005). Given a sequence of words $W = \{w_1, w_2 ... w_n\}$, each word w_i with several senses $\{s_{i1}, s_{i2} ... s_{im}\}$. A graph G = (V,E) is defined such that there exists a vertex v for each sense. Two senses of two different words may be connected by an edge e, depending on their distance. That two senses are connected suggests they should have influence on each other, accordingly a maximum

[2] http://wordnet.princeton.edu

[3] We acknowledge the existence of many research papers that tackled the AW WSD problem using unsupervised approaches, yet for lack of space we will not be able to review most of them.

allowable distance is set. They explore 4 different graph based algorithms. The highest yielding algorithm in their work is the In-Degree algorithm combining different WN similarity measures depending on POS. They used the Jiang and Conrath (JCN) (Jiang and Conrath., 1997) similarity measure within nouns, the Leacock & Chodorow (LCH) (Leacock and Chodorow, 1998) similarity measure within verbs, and the Lesk (Lesk, 1986) similarity measure within adjectives, within adverbs, and among different POS tag pairings. They evaluate their work against the SENSEVAL 2 AW test data (SV2AW). They tune the parameters of their algorithm – namely, the normalization ratio for some of these measures – on the SENSEVAL 3 data set. They report a state-of-the-art unsupervised system that yields an overall performance across all AW POS sets of 57.2%.

In our current work, we extend the SM07 work in some interesting ways. A detailed narrative of our approach is described in (Guo and Diab, 2009). Briefly, we focus on the In-Degree graph based algorithm since it is the best performer in the SM07 work. The In-Degree algorithm presents the problem as a weighted graph with senses as nodes and the similarity between senses as weights on edges. The In-Degree of a vertex refers to the number of edges incident on that vertex. In the weighted graph, the In-Degree for each vertex is calculated by summing the weights on the edges that are incident on it. After all the In-Degree values for each sense are computed, the sense with maximum value is chosen as the final sense for that word.

In this paper, we use the In-Degree algorithm while applying some modifications to the basic similarity measures exploited and the WN lexical resource tapped into. Similar to the original In-Degree algorithm, we produce a probabilistic ranked list of senses. Our modifications are described as follows:

JCN for Verb-Verb Similarity In our implementation of the In-Degree algorithm, we use the JCN similarity measure for both Noun-Noun similarity calculation similar to SM07. However, different from SM07, instead of using LCH for Verb-Verb similarity, we use the JCN metric as it yields better performance in our experimentations.

Expand Lesk Following the intuition in (Pedersen et al., 2005), henceforth (PEA05), we expand the basic Lesk similarity measure to take into account the glosses for all the relations for the synsets on the contextual words and compare them with the glosses of the target word senses, therefore going beyond the is-a relation. We exploit the observation that WN senses are too fine-grained, accordingly the neighbors would be slightly varied while sharing significant semantic meaning content. To find similar senses, we use the relations: hypernym, hyponym, similar attributes, similar verb group, pertinym, holonym, and meronyms.[4] The algorithm assumes that the words in the input are POS tagged. In PEA05, the authors retrieve all the relevant neighbors to form a bag of words for both the target sense and the surrounding senses of the context words, they specifically focus on the Lesk similarity measure. In our current work, we employ the neighbors in a disambiguation strategy using different similarity measures one pair at a time. Our algorithm takes as input a target sense and a sense pertaining to a word in the surrounding context, and returns a sense similarity score. We do not apply the WN relations expansion to the target sense. It is only applied to the contextual word.[5]

For the monolingual system, we employ the same normalization values used in SM07 for the different similarity measures. Namely for the Lesk and Expand-Lesk, we use the same cut-off value of 240, accordingly, if the Lesk or Expand-Lesk similarity value returns $0 <= 240$ it is converted to a real number in the interval [0,1], any similarity over 240 is by default mapped to 1. We will refer to the Expand-Lesk with this threshold as Lesk2. We also experimented with different thresholds for the Lesk and Expand-Lesk similarity measure using the SENSEVAL 3 data as a tuning set. We found that a cut-off threshold of 40 was also useful. We will refer to this variant of Expand-Lesk with a cut off threshold of 40 as Lesk3. For JCN, similar to SM07, the values are from 0.04 to 0.2, we mapped them to the interval [0,1]. We did not run any calibration studies beyond the what was reported in SM07.

[4] In our experiments, we varied the number of relations to employ and they all yielded relatively similar results. Hence in this paper, we report results using all the relations listed above.

[5] We experimented with expanding both the contextual sense and the target sense and we found that the unreliability of some of the relations is detrimental to the algorithm's performance. Hence we decided empirically to expand only the contextual word.

SemCor Expansion of WN A part of the `RelCont` approach relies on using the Lesk algorithm. Accordingly, the availability of glosses associated with the WN entries is extremely beneficial. Therefore, we expand the number of glosses available in WN by using the SemCor data set, thereby adding more examples to compare. The SemCor corpus is a corpus that is manually sense tagged (Miller, 1990).[6] In this expansion, depending on the version of WN, we use the sense-index file in the WN Database to convert the SemCor data to the appropriate version sense annotations. We augment the sense entries for the different POS WN databases with example usages from SemCor. The augmentation is done as a look up table external to WN proper since we did not want to dabble with the WN offsets. We set a cap of 30 additional examples per synset. We used the first 30 examples with no filtering criteria. Many of the synsets had no additional examples. WN1.7.1 comprises a total of 26875 synsets, of which 25940 synsets are augmented with SemCor examples.[7]

4.2 Multilingual System `TransCont`

`TransCont` is based on the WSD system SALAAM (Diab and Resnik, 2002), henceforth (DR02). The SALAAM system leverages word alignments from parallel corpora to perform WSD. The SALAAM algorithm exploits the word correspondence cross linguistically to tag word senses on words in running text. It relies on several underlying assumptions. The first assumption is that senses of polysemous words in one language could be lexicalized differently in other languages. For example, 'bank' in English would be translated as *banque* or *rive de fleuve* in French, depending on context. The other assumption is that if Language 1 (L1) words are translated to the same orthographic form in Language 2 (L2), then they share the some element of meaning, they are semantically similar.[8]

The SALAAM algorithm can be described as follows. Given a parallel corpus of L1-L2 that is sentence and word aligned, group all the word types in L1 that map to same word in L2 creating clusters referred to as *t*ypesets. Then perform disambiguation on the typeset clusters using WN. Once senses are identified for each word in the cluster, the senses are propagated back to the original word instances in the corpus. In the SALAAM algorithm, the disambiguation step is carried out as follows: within each of these target sets consider all possible sense tags for each word and choose sense tags informed by semantic similarity with all the other words in the whole group. The algorithm is a greedy algorithm that aims at maximizing the similarity of the chosen sense across all the words in the set. The SALAAM disambiguation algorithm used the noun groupings (NounGroupings) algorithm described in DR02. The algorithm applies disambiguation within POS tag. The authors report only results on the nouns only since NounGroupings heavily exploits the hierarchy structure of the WN noun taxonomy, which does not exist for adjectives and adverbs, and is very shallow for verbs.

Essentially SALAAM relies on variability in translation as it is important to have multiple words in a typeset to allow for disambiguation. In the original SALAAM system, the authors automatically translated several balanced corpora in order to render more variable data for the approach to show it's impact. The corpora that were translated are: the WSJ, the Brown corpus and all the SENSEVAL data. The data were translated to different languages (Arabic, French and Spanish) using state of art MT systems. They employed the automatic alignment system GIZA++ (Och and Ney, 2003) to obtain word alignments in a single direction from L1 to L2.

For `TransCont` we use the basic SALAAM approach with some crucial modifications that lead to better performance. We still rely on parallel corpora, we extract typesets based on the intersection of word alignments in both alignment directions using more advanced GIZA++ machinery. In contrast to DR02, we experiment with all four POS: Verbs (V), Nouns (N), Adjectives (A) and Adverbs (R). Moreover, we modified the underlying disambiguation method on the typesets. We still employ WN similarity, however, we do not use the NounGroupings algorithm. Our disambiguation method relies on calculating the sense pair similarity exhaustively across all the

[6] Using SemCor in this setting to augment WN does hint of using supervised data in the WSD process, however, since our approach does not rely on training data and SemCor is not used in our algorithm directly to tag data, but to augment a rich knowledge resource, we contend that this does not affect our system's designation as an unsupervised system.

[7] Some example sentences are repeated across different synsets and POS since the SemCor data is annotated as an All-Words tagged data set.

[8] We implicitly make the underlying simplifying assumption that the L2 words are less ambiguous than the L1 words.

word types in a typeset and choosing the combination that yields the highest similarity. We experimented with all the WN similarity measures in the WN similarity package.[9] We also experiment with Lesk2 and Lesk3 as well as other measures, however we do not use SemCor examples with `TransCont`. We found that the best results are yielded using the Lesk2/Lesk3 similarity measure for N, A and R POS tagsets, while the Lin and JCN measures yield the best performance for the verbs. In contrast to the DR02 approach, we modify the internal WSD process to use the `In-Degree` algorithm on the typeset, so each sense obtains a confidence, and the sense(s) with the highest confidences are returned.

4.3 Combining `RelCont` and `TransCont`

Our objective is to combine the different sources of evidence for the purposes of producing an effective overall global WSD system that is able to disambiguate all content words in running text. We combine the two systems in two different ways.

4.3.1 MERGE

In this combination scheme, the words in the typeset that result from the `TransCont` approach are added to the context of the target word in the `RelCont` approach. However the typeset words are not treated the same as the words that come from the surrounding context in the `In-Degree` algorithm as we recognize that words that are yielded in the typesets are semantically similar in terms of content rather than being co-occurring words as is the case for contextual words in RelCont. Heeding this difference, we proceed to calculate similarity for words in the typesets using different similarity measures. In the case of noun-noun similarity, in the original `RelCont` experiments we use JCN, however with the words present in the `TransCont` typesets we use one of the Lesk variants, Lesk2 or Lesk3. Our observation is that the JCN measure is relatively coarser grained, compared to Lesk measures, therefore it is sufficient in case of lexical relatedness therefore works well in case of the context words. Yet for the words yielded in the `TransCont` typesets a method that exploits the underlying rich relations in the noun hierarchy captures the semantic similarity more aptly. In the case of verbs we still maintain the JCN similarity as it most effective given the shallowness of the verb hierarchy and the inherent nature of the verbal synsets which are differentiated along syntactic rather than semantic dimensions. We employ the Lesk algorithm still with A-A and R-R similarity and when comparing across different POS tag pairings.

4.3.2 VOTE

In this combination scheme, the output of the global disambiguation system is simply an intersection of the two outputs from the two underlying systems `RelCont` and `TransCont`. Specifically, we sum up the confidence ranging from 0 to 1 of the two system `In-Degree` algorithm outputs to obtain a final confidence for each sense, choosing the sense(s) that yields the highest confidences. The fact that `TransCont` uses `In-Degree` internally allows for a seamless integration.

5 Experiments and Results

5.1 Data

The parallel data we experiment with are the same standard data sets as in (Diab and Resnik, 2002), namely, Senseval 2 English AW data sets (SV2AW) (Palmer et al., 2001), and Seneval 3 English AW (SV3AW) data set. We use the true POS tag sets in the test data as rendered in the Penn Tree Bank.[10] We present our results on WordNet 1.7.1 for ease of comparison with previous results.

5.2 Evaluation Metrics

We use the `scorer2` software to report fine-grained (P)recision and (R)ecall and (F)-measure.

5.3 Baselines

We consider here several baselines. 1. A random baseline (RAND) is the most appropriate baseline for an unsupervised approach.2. We include the most frequent sense baseline (MFBL), though we note that we consider the most frequent sense or first sense baseline to be a supervised baseline since it depends crucially on SemCor in ranking the senses within WN.[11] 3. The SM07 results as a

[9] http://wn-similarity.sourceforge.net/

[10] We exclude the data points that have a tag of "U" in the gold standard for both baselines and our system.

[11] From an application standpoint, we do not find the first sense baseline to be of interest since it introduces a strong level of uniformity – removing semantic variability – which is not desirable. Even if the first sense achieves higher results in data sets, it is an artifact of the size of the data and the very limited number of documents under investigation.

monolingual baseline. 4. The DR02 results as the multilingual baseline.

5.4 Experimental Results

5.4.1 RelCont

We present the results for 4 different experimental conditions for RelCont: JCN-V which uses JCN instead of LCH for verb-verb similarity comparison, we consider this our base condition; +ExpandL is adding the Lesk Expansion to the base condition, namely Lesk2;[12] +SemCor adds the SemCor expansion to the base condition; and finally +ExpandL_SemCor, adds the latter both conditions simultaneously. Table 1 illustrates the obtained results for the SV2AW using WordNet 1.7.1 since it is the most studied data set and for ease of comparison with previous studies. We break the results down by POS tag (N)oun, (V)erb, (A)djective, and Adve(R)b. The coverage for SV2AW is 98.17% losing some of the verb and adverb target words.

Our overall results on all the data sets clearly outperform the baseline as well as state-of-the-art performance using an unsupervised system (SM07) in overall f-measure across all the data sets. We are unable to beat the most frequent baseline (MFBL) which is obtained using the first sense. However MFBL is a supervised baseline and our approach is unsupervised. Our implementation of SM07 is slightly higher than those reported in (Sinha and Mihalcea, 2007) (57.12%) is probably due to the fact that we do not consider the items tagged as "U" and also we resolve some of the POS tag mismatches between the gold set and the test data. We note that for the SV2AW data set our coverage is not 100% due to some POS tag mismatches that could not have been resolved automatically. These POS tag problems have to do mainly with multiword expressions. In observing the performance of the overall RelCont, we note that using JCN for verbs clearly outperforms using the LCH similarity measure. Using SemCor to augment WN examples seems to have the biggest impact. Combining SemCor with ExpandL yields the best results.

Observing the results yielded per POS in Table 1, ExpandL seems to have the biggest impact on the Nouns only. This is understandable since the noun hierarchy has the most dense relations and the most consistent ones. SemCor augmentation of WN seemed to benefit all POS significantly except for nouns. In fact the performance on the nouns deteriorated from the base condition JCN-V from 68.7 to 68.3%. This maybe due to inconsistencies in the annotations of nouns in SemCor or the very fine granularity of the nouns in WN. We know that 72% of the nouns, 74% of the verbs, 68.9% of the adjectives, and 81.9% of the adverbs directly exploited the use of SemCor augmented examples. Combining SemCor and ExpandL seems to have a positive impact on the verbs and adverbs, but not on the nouns and adjectives. These trends are not held consistently across data sets. For example, we see that SemCor augmentation helps all POS tag sets over using ExpandL alone or even when combined with SemCor. We note the similar trends in performance for the SV3AW data.

Compared to state of the art systems, RelCont with an overall F-measure performance of 62.13% outperforms the best unsupervised system of 57.5% UNED-AW-U2 for SV2 (Navigli, 2009). It is worth noting that it is higher than several of the supervised systems. Moreover, RelCont yields better overall results on SV3 at 59.87 compared to the best unsupervised system IRST-DDD-U which yielded an F-measure of 58.3% (Navigli, 2009).

5.4.2 TransCont

For the TransCont results we illustrate the original SALAAM results as our baseline. Similar to the DR02 work, we actually use the same SALAAM parallel corpora comprising more than 5.5M English tokens translated using a single machine translation system GlobalLink. Therefore our parallel corpus is the French English translation condition mentioned in DR02 work as FrGl. We have 4 experimental conditions: FRGL using Lesk2 for all POS tags in the typeset disambiguation (Lesk2); FRGL using Lesk3 for all POS tags (Lesk3); using Lesk3 for N, A and R but LIN similarity measure for verbs (Lesk3_Lin); using Lesk3 for N, A and R but JCN for verbs (Lesk3_JCN).

In Table 3 we note the the Lesk3_JCN followed immediately by Lesk3_Lin yield the best performance. The trend holds for both SV2AW and SV3AW. Essentially our new implementation of the multilingual system significantly outperforms the original DR02 implementation for all experimental conditions.

[12] Using Lesk3 yields almost the same results

Condition	N	V	A	R	Global F Measure
RAND	43.7	21	41.2	57.4	39.9
MFBL	71.8	41.45	67.7	81.8	65.35
SM07	68.7	33.01	65.2	63.1	59.2
JCN-V	68.7	35.46	65.2	63.1	59.72
+ExpandL	**70.2**	35.86	65.4	62.45	60.48
+SemCor	68.5	**38.66**	**69.2**	67.75	61.79
+ExpandL_SemCor	69.0	**38.66**	68.8	**69.45**	**62.13**

Table 1: `RelCont` F-measure results per POS tag per condition for SV2AW using WN 1.7.1.

Condition	N	V	A	R	Global F Measure
RAND	39.67	19.34	41.85	92.31	32.97
MFBL	70.4	54.15	66.7	92.88	63.96
SM07	60.9	43.4	57	92.88	53.98
JCN-V	60.9	48.5	57	92.88	55.87
+ExpandL	59.9	48.55	57.95	92.88	55.62
+SemCor	**66**	48.95	**65.55**	92.88	**59.87**
+ExpandL_SemCor	65	**49.2**	**65.55**	92.88	59.52

Table 2: `RelCont` F-measure results per POS tag per condition for SV3AW using WN 1.7.1.

5.4.3 Global Combined WSD

In this section we present the results of the global combined WSD system. All the combined experimental conditions have the same percentage coverage.[13] We present the results combining using MERGE and using VOTE. We have chosen 4 baseline systems: (1) SM07; (2) the our baseline monolingual system using JCN for verb-verb comparisons (RelCont-BL), so as to distinguish the level of improvement that could be attributed to the multilingual system in the combination results; as well as (3) and (4) our best individual system results from `RelCont` (ExpandL_SemCor) referred to in the tables below as (RelCont-Final) and `TransCont` using the best experimental condition (Lesk3_JCN). Table 5 and 6 illustrates the overall performance of our combined approach.

In Table 5 we note that the combined conditions outperform the two base systems independently, using `TransCont` is always helpful for any of the 3 monolingual systems, no matter we use VOTE or MERGE. In general the trend is that VOTE outperforms MERGE, however they exhibit different behaviors with respect to what works for each POS.

In Table 6 the combined result is not always better than the corresponding monolingual system. When applying to our baseline monolingual system, the combined result is still better. However, we observed worse results for ExpandL_Semcor, RelCont-Final. There may be 2 main reasons for the loss: (1) SV3 is the tuning set in SM07, and we inherit the thresholds for similarity metrics from that study. Accordingly, an overfitting of the thresholds is probably happening in this case; (2) `TransCont` results are not good enough on the SV3AW data. Comparing the `RelCont` and `TransCont` system results, we find a drop in f-measure of -1.37% in SV2AW, in contrast to a much larger drop in performance for the SV3AW data set where the drop in performance is -6.38% when comparing `RelCont-BL` to `TransCont` and nearly -10% comparing against `RelCont-Final`.

6 Discussion

We looked closely at the data in the combined conditions attempting to get a feel for the data and understand what was captured and what was not. Some of the good examples that are captured in the combined system that are not tagged in `RelCont` is the case of *ringer* in *Like most of the other 6,000 churches in Britain with sets of bells , St. Michael once had its own " band " of ringers , who would herald every Sunday morning and evening service* .. The `RelCont` answer is *ringer* sense number 4: *(horseshoes) the successful throw of a horseshoe*

[13] We do not back off in any of our systems to a default sense, hence the coverage is not at a 100%.

Condition	N	V	A	R	Global F Measure
RAND	43.7	21	41.2	57.4	39.9
DR02-FRGL	54.5				
SALAAM	65.48	31.77	56.87	67.4	57.23
Lesk2	67.05	30	59.69	68.01	57.27
Lesk3	67.15	30	60.2	68.01	57.41
Lesk3_Lin	67.15	29.27	**60.2**	**68.01**	57.61
Lesk3_JCN	**67.15**	**33.88**	**60.2**	**68.01**	**58.35**

Table 3: `TransCont` F-measure results per POS tag per condition for SV2AW using WN 1.7.1.

Condition	N	V	A	R	Global F Measure
RAND	39.67	19.34	41.85	92.31	32.93
SALAAM	52.42	29.27	54.14	88.89	45.63
Lesk2	53.57	33.58	53.63	88.89	47
Lesk3	53.77	33.30	56.48	88.89	47.5
Lesk3_Lin	53.77	29.24	56.48	88.89	46.37
Lesk3_JCN	53.77	38.43	56.48	88.89	**49.29**

Table 4: `TransCont` F-measure results per POS tag per condition for SV3AW using WN 1.7.1.

or quoit so as to encircle a stake or peg. When the merged system is employed we see the correct sense being chosen as sense number 1 in the MERGE condition: defined in WN as *a person who rings church bells (as for summoning the congregation)* resulting from a corresponding translation into French as *s*onneur.

We did some basic data analysis on the items we are incapable of capturing. Several of them are cases of metonymy in examples such as "the English are known...", the sense of *English* here is clearly in reference to the people of England, however, our WSD system preferred the language sense of the word. These cases are not gotten by any of our systems. If it had access to syntactic/semantic roles we assume it could capture that this sense of the word entails volition for example. Other types of errors resulted from the lack of a way to explicitly identify multiwords.

Looking at the performance of `TransCont` we note that much of the loss is a result of the lack of variability in the translations which is a key factor in the performance of the algorithm. For example for the 157 adjective target test words in SV2AW, there was a single word alignment for 51 of the cases, losing any tagging for these words.

7 Conclusions and Future Directions

In this paper we present a framework that combines orthogonal sources of evidence to create a state-of-the-art system for the task of WSD disambiguation for AW. Our approach yields an overall global F measure of 64.58 for the standard SV2AW data set combining monolingual and multilingual evidence. The approach can be further refined by adding other types of orthogonal features such as syntactic features and semantic role label features. Adding SemCor examples to `TransCont` should have a positive impact on performance. Also adding more languages as illustrated by the DR02 work should also yield much better performance.

References

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, Prague, Czech Republic, June. Association for Computational Linguistics.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic, June. Association for Computational Linguistics.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*,

Condition	N	V	A	R	Global F Measure
SM07	68.7	33.01	65.2	63.1	59.2
RelCont-BL	68.7	35.46	65.2	63.1	59.72
RelCont-Final	69.0	38.66	68.8	69.45	**62.13**
TransCont	67.15	33.88	60.2	68.01	58.35
MERGE: RelCont-BL+TransCont	69.3	36.91	66.7	64.45	60.82
VOTE: RelCont-BL+TransCont	71	37.71	66.5	66.1	61.92
MERGE: RelCont-Final+TransCont	70.7	38.66	69.5	70.45	63.14
VOTE: RelCont-Final+TransCont	74.2	38.26	68.6	71.45	**64.58**

Table 5: F-measure % for all Combined experimental conditions on SV2AW

Condition	N	V	A	R	Global F Measure
SM07	60.9	43.4	57	92.88	53.98
RelCont-BL	60.9	48.5	57	92.88	55.87
RelCont-Final	65	49.2	65.55	92.88	**59.52**
TransCont	53.77	38.43	56.48	88.89	49.29
MERGE: RelCont-BL+TransCont	60.6	49.5	58.85	92.88	56.47
VOTE: RelCont-BL+TransCont	59.3	49.5	59.1	92.88	55.92
MERGE: RelCont-Final+TransCont	63.2	50.3	65.25	92.88	**59.07**
VOTE: RelCont-Final+TransCont	62.4	49.65	65.25	92.88	58.47

Table 6: F-measure % for all Combined experimental conditions on SV3AW

pages 255–262, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Christiane Fellbaum. 1998. "wordnet: An electronic lexical database". MIT Press.

Weiwei Guo and Mona Diab. 2009. Improvements to monolingual english word sense disambiguation. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 64–69, Boulder, Colorado, June. Association for Computational Linguistics.

N. Ide and J. Veronis. 1998. Word sense disambiguation: The state of the art. In *Computational Linguistics*, pages 1–40, 24:1.

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

C. Leacock and M. Chodorow. 1998. Combining local context and wordnet sense similarity for word sense identification. In *WordNet, An Electronic Lexical Database*. The MIT Press.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *In Proceedings of the SIGDOC Conference*, Toronto, June.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 411–418, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

George A. Miller. 1990. Wordnet: a lexical database for english. In *Communications of the ACM*, pages 39–41.

Roberto Navigli and Mirella Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the* 20^{th} *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1683–1688, Hyderabad, India.

Roberto Navigli. 2009. Word sense disambiguation: a survey. In *ACM Computing Surveys*, pages 1–69. ACM Press.

Franz Joseph Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

M. Palmer, C. Fellbaum, S. Cotton, L. Delfs, , and H. Dang. 2001. English tasks: all-words and verb lexical sample. In *In Proceedings of ACL/SIGLEX Senseval-2*, Toulouse, France, June.

Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. In *University of Minnesota Supercomputing Institute Research Report UMSI 2005/25*, Minnesotta, March.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task-17: English lexical sample, srl and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic, June. Association for Computational Linguistics.

Ravi Sinha and Rada Mihalcea. 2007. Unsupervised graph-based word sense disambiguation using measures of word semantic similarity. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007)*, Irvine, CA.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July. Association for Computational Linguistics.

Phrase-based Statistical Language Generation using Graphical Models and Active Learning

François Mairesse, **Milica Gašić**, **Filip Jurčíček**,
Simon Keizer, **Blaise Thomson**, **Kai Yu** and **Steve Young***
Cambridge University Engineering Department, Trumpington Street, Cambridge, CB2 1PZ, UK
{f.mairesse, mg436, fj228, sk561, brmt2, ky219, sjy}@eng.cam.ac.uk

Abstract

Most previous work on trainable language generation has focused on two paradigms: (a) using a statistical model to rank a set of generated utterances, or (b) using statistics to inform the generation decision process. Both approaches rely on the existence of a handcrafted generator, which limits their scalability to new domains. This paper presents BAGEL, a statistical language generator which uses dynamic Bayesian networks to learn from semantically-aligned data produced by 42 untrained annotators. A human evaluation shows that BAGEL can generate natural and informative utterances from *unseen* inputs in the information presentation domain. Additionally, generation performance on sparse datasets is improved significantly by using certainty-based active learning, yielding ratings close to the human gold standard with a fraction of the data.

1 Introduction

The field of natural language generation (NLG) is one of the last areas of computational linguistics to embrace statistical methods. Over the past decade, statistical NLG has followed two lines of research. The first one, pioneered by Langkilde and Knight (1998), introduces statistics in the generation process by training a model which reranks candidate outputs of a handcrafted generator. While their HALOGEN system uses an n-gram language model trained on news articles, other systems have used hierarchical syntactic models (Bangalore and Rambow, 2000), models trained on user ratings of utterance quality (Walker et al., 2002), or alignment models trained on speaker-specific corpora (Isard et al., 2006).

A second line of research has focused on introducing statistics at the generation decision level, by training models that find the set of generation parameters maximising an objective function, e.g. producing a target linguistic style (Paiva and Evans, 2005; Mairesse and Walker, 2008), generating the most likely context-free derivations given a corpus (Belz, 2008), or maximising the expected reward using reinforcement learning (Rieser and Lemon, 2009). While such methods do not suffer from the computational cost of an overgeneration phase, they still require a handcrafted generator to define the generation decision space within which statistics can be used to find an optimal solution.

This paper presents BAGEL (Bayesian networks for generation using active learning), an NLG system that can be fully trained from aligned data. While the main requirement of the generator is to produce natural utterances within a dialogue system domain, a second objective is to minimise the overall development effort. In this regard, a major advantage of data-driven methods is the shift of the effort from model design and implementation to data annotation. In the case of NLG systems, learning to produce paraphrases can be facilitated by collecting data from a large sample of annotators. Our meaning representation should therefore (a) be intuitive enough to be understood by untrained annotators, and (b) provide useful generalisation properties for generating unseen inputs. Section 2 describes BAGEL's meaning representation, which satisfies both requirements. Section 3 then details how our meaning representation is mapped to a phrase sequence, using a dynamic Bayesian network with backoff smoothing.

Within a given domain, the same semantic concept can occur in different utterances. Section 4 details how BAGEL exploits this redundancy

This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and funded by the EU FP7 Programme under grant agreement 216594 (CLASSiC project: www.classic-project.org).

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1552–1561,
Uppsala, Sweden, 11-16 July 2010.

to improve generation performance on sparse datasets, by guiding the data collection process using *certainty-based active learning* (Lewis and Catlett, 1994). We train BAGEL in the information presentation domain, from a corpus of utterances produced by 42 untrained annotators (see Section 5.1). An automated evaluation metric is used to compare preliminary model and training configurations in Section 5.2, while Section 5.3 shows that the resulting system produces natural and informative utterances, according to 18 human judges. Finally, our human evaluation shows that training using active learning significantly improves generation performance on sparse datasets, yielding results close to the human gold standard using a fraction of the data.

2 Phrase-based generation from semantic stacks

BAGEL uses a *stack-based* semantic representation to constrain the sequence of semantic concepts to be searched. This representation can be seen as a linearised semantic tree similar to the one previously used for natural language understanding in the Hidden Vector State model (He and Young, 2005). A stack representation provides useful generalisation properties (see Section 3.1), while the resulting stack sequences are relatively easy to align (see Section 5.1). In the context of dialogue systems, Table 1 illustrates how the input dialogue act is first mapped to a set of stacks of semantic concepts, and then aligned with a word sequence. The bottom concept in the stack will typically be a *dialogue act type*, e.g. an utterance providing information about the object under discussion (`inform`) or specifying that the request of the user cannot be met (`reject`). Other concepts include attributes of that object (e.g., `food`, `area`), values for those attributes (e.g., `Chinese`, `riverside`), as well as special symbols for negating underlying concepts (e.g., `not`) or specifying that they are irrelevant (e.g., `dontcare`).

The generator's goal is thus finding the most likely realisation given an *unordered* set of *mandatory* semantic stacks $\mathcal{S}_m$ derived from the input dialogue act. For example, $s =$`inform(area(centre))` is a mandatory stack associated with the dialogue act in Table 1 (frame 8). While mandatory stacks must all be conveyed in the output realisation, $\mathcal{S}_m$ does not contain the optional *intermediary* stacks $\mathcal{S}_i$ that can refer to (a) general attributes of the object under discussion (e.g., `inform(area)` in Table 1), or (b) to concepts that are not in the input at all, which are associated with the singleton stack `inform` (e.g., phrases expressing the dialogue act type, or clause aggregation operations). For example, the stack sequence in Table 1 contains 3 intermediary stacks for $t = 2, 5$ and 7.

BAGEL's granularity is defined by the semantic annotation in the training data, rather than external linguistic knowledge about what constitutes a unit of meaning, i.e. contiguous words belonging to the same semantic stack are modelled as an atomic observation unit or *phrase*.[1] In contrast with word-level models, a major advantage of phrase-based generation models is that they can model long-range dependencies and domain-specific idiomatic phrases with fewer parameters.

3 Dynamic Bayesian networks for NLG

Dynamic Bayesian networks have been used successfully for speech recognition, natural language understanding, dialogue management and text-to-speech synthesis (Rabiner, 1989; He and Young, 2005; Lefèvre, 2006; Thomson and Young, 2010; Tokuda et al., 2000). Such models provide a principled framework for predicting elements in a large structured space, such as required for non-trivial NLG tasks. Additionally, their probabilistic nature makes them suitable for modelling linguistic variation, i.e. there can be multiple valid paraphrases for a given input.

BAGEL models the generation task as finding the most likely sequence of realisation phrases $\mathbf{R}^* = (r_1 ... r_L)$ given an unordered set of mandatory semantic stacks $\mathcal{S}_m$, with $|\mathcal{S}_m| \leq L$. BAGEL must thus derive the optimal sequence of semantic stacks $\mathbf{S}^*$ that will appear in the utterance given $\mathcal{S}_m$, i.e. by inserting intermediary stacks if needed and by performing content ordering. Any number of intermediary stacks can be inserted between two consecutive mandatory stacks, as long as all their concepts are included in either the previous or following mandatory stack, and as long as each stack transition leads to a different stack (see example in Table 1). Let us define the set of possible stack sequences matching these constraints as $Seq(\mathcal{S}_m) \subseteq \{\mathbf{S} = (s_1 ... s_L) \text{ s.t. } s_t \in \mathcal{S}_m \cup \mathcal{S}_i\}$.

We propose a model which estimates the dis-

[1] The term *phrase* is thus defined here as any sequence of one or more words.

Charlie Chan	*is a*	*Chinese*	*restaurant*	*near*	*Cineworld*	*in the*	*centre of town*
Charlie Chan name inform	`inform`	**Chinese food inform**	**restaurant type inform**	`near inform`	**Cineworld near inform**	`area inform`	**centre area inform**
$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$

Table 1: Example semantic stacks aligned with an utterance for the dialogue act `inform(name(Charlie Chan) type(restaurant) area(centre) food(Chinese) near(Cineworld))`. Mandatory stacks are in bold.

tribution $P(\mathbf{R}|\mathcal{S}_m)$ from a training set of realisation phrases aligned with semantic stack sequences, by marginalising over all stack sequences in $Seq(\mathcal{S}_m)$:

$$\begin{aligned} P(\mathbf{R}|\mathcal{S}_m) &= \sum_{\mathbf{S}\in Seq(\mathcal{S}_m)} P(\mathbf{R},\mathbf{S}|\mathcal{S}_m) \\ &= \sum_{\mathbf{S}\in Seq(\mathcal{S}_m)} P(\mathbf{R}|\mathbf{S},\mathcal{S}_m)P(\mathbf{S}|\mathcal{S}_m) \\ &= \sum_{\mathbf{S}\in Seq(\mathcal{S}_m)} P(\mathbf{R}|\mathbf{S})P(\mathbf{S}|\mathcal{S}_m) \qquad (1) \end{aligned}$$

Inference over the model defined in (1) requires the decoding algorithm to consider all possible orderings over $Seq(\mathcal{S}_m)$ together with all possible realisations, which is intractable for non-trivial domains. We thus make the additional assumption that the most likely sequence of semantic stacks $\mathbf{S}^*$ given $\mathcal{S}_m$ is the one yielding the optimal realisation phrase sequence:

$$P(\mathbf{R}|\mathcal{S}_m) \approx P(\mathbf{R}|\mathbf{S}^*)P(\mathbf{S}^*|\mathcal{S}_m) \qquad (2)$$

$$\text{with } \mathbf{S}^* = \operatorname*{argmax}_{\mathbf{S}\in Seq(\mathcal{S}_m)} P(\mathbf{S}|\mathcal{S}_m) \qquad (3)$$

The semantic stacks are therefore decoded first using the model in Fig. 1 to solve the argmax in (3). The decoded stack sequence $\mathbf{S}^*$ is then treated as observed in the realisation phase, in which the model in Fig. 2 is used to find the realisation phrase sequence $\mathbf{R}^*$ maximising $P(\mathbf{R}|\mathbf{S}^*)$ over all phrase sequences of length $L = |\mathbf{S}^*|$ in our vocabulary:

$$\begin{aligned} \mathbf{R}^* &= \operatorname*{argmax}_{\mathbf{R}=(r_1\ldots r_L)} P(\mathbf{R}|\mathbf{S}^*)P(\mathbf{S}^*|\mathcal{S}_m) \qquad (4) \\ &= \operatorname*{argmax}_{\mathbf{R}=(r_1\ldots r_L)} P(\mathbf{R}|\mathbf{S}^*) \qquad (5) \end{aligned}$$

In order to reduce model complexity, we factorise our model by conditioning the realisation phrase at time t on the previous phrase r_{t-1}, and the previous, current, and following semantic stacks. The semantic stack s_t at time t is assumed

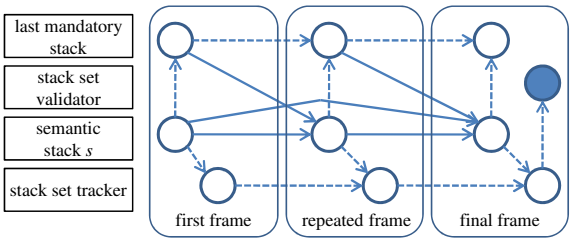


Figure 1: Graphical model for the semantic decoding phase. Plain arrows indicate smoothed probability distributions, dashed arrows indicate deterministic relations, and shaded nodes are observed. The generation of the `end` semantic stack symbol deterministically triggers the final frame.

to depend only on the previous two stacks and the last mandatory stack $s_u \in \mathcal{S}_m$ with $1 \le u < t$:

$$P(\mathbf{S}|\mathcal{S}_m) = \begin{cases} \prod_{t=1}^{T} P(s_t|s_{t-1}, s_{t-2}, s_u) \\ \quad \text{if } \mathbf{S} \in Seq(\mathcal{S}_m) \\ 0 \quad \text{otherwise} \end{cases} \qquad (6)$$

$$P(\mathbf{R}|\mathbf{S}^*) = \prod_{t=1}^{T} P(r_t|r_{t-1}, s^*_{t-1}, s^*_t, s^*_{t+1}) \qquad (7)$$

While dynamic Bayesian networks typically take sequential inputs, mapping a *set* of semantic stacks to a sequence of phrases is achieved by keeping track of the mandatory stacks that were visited in the current sequence (see stack set tracker variable in Fig. 1), and pruning any sequence that has not included all mandatory input stacks on reaching the final frame (see observed stack set validator variable in Fig. 1). Since the number of intermediary stacks is not known at decoding time, the network is unrolled for a fixed number of frames T defining the maximum number of phrases that can be generated (e.g., $T = 50$). The end of the stack sequence is then determined by a special `end` symbol, which can only be emitted within the T frames once all mandatory stacks have been visited. The probability of the resulting utterance is thus computed over all frames up to the `end` symbol, which determines the length

L of $\mathbf{S}^*$ and $\mathbf{R}^*$. While the decoding constraints enforce that $L > |\mathcal{S}_m|$, the search for $\mathbf{S}^*$ requires comparing sequences of different lengths. A consequence is that shorter sequences containing only mandatory stacks are likely to be favoured. While future work should investigate length normalisation strategies, we find that the learned transition probabilities are skewed enough to favour stack sequences including intermediary stacks.

Once the topology and the decoding constraints of the network have been defined, any inference algorithm can be used to search for $\mathbf{S}^*$ and $\mathbf{R}^*$. We use the junction tree algorithm implemented in the Graphical Model ToolKit (GMTK) for our experiments (Bilmes and Zweig, 2002), however both problems can be solved using a standard Viterbi search given the appropriate state representation. In terms of computational complexity, it is important to note that the number of stack sequences $Seq(\mathcal{S}_m)$ to search over increases exponentially with the number of input mandatory stacks. Nevertheless, we find that real-time performance can be achieved by pruning low probability sequences, without affecting the quality of the solution.

3.1 Generalisation to unseen semantic stacks

In order to generalise to semantic stacks which have not been observed during training, the realisation phrase r is made dependent on underspecified stack configurations, i.e. the tail l and the head h of the stack. For example, the last stack in Table 1 is associated with the head `centre` and the tail `inform(area)`. As a result, BAGEL assigns non-zero probabilities to realisation phrases in unseen semantic contexts, by backing off to the head and the tail of the stack. A consequence is that BAGEL's lexical realisation can generalise across contexts. For example, if `reject(area(centre))` was never observed at training time, $P(r = \text{centre of town}|s = \texttt{reject(area(centre))})$ will be estimated by backing off to $P(r = \text{centre of town}|h = \texttt{centre})$. BAGEL can thus generate 'there are no venues in the centre of town' if the phrase 'centre of town' was associated with the concept `centre` in a different context, such as `inform(area(centre))`. The final realisation model is illustrated in Fig. 2:

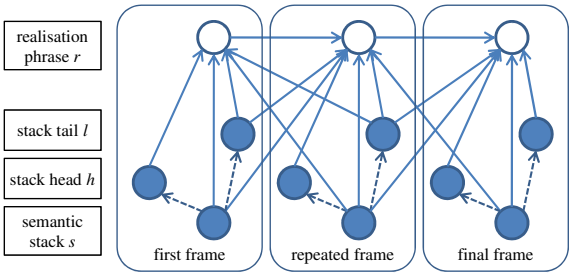


Figure 2: Graphical model for the realisation phase. Dashed arrows indicate deterministic relations, and shaded node are observed.

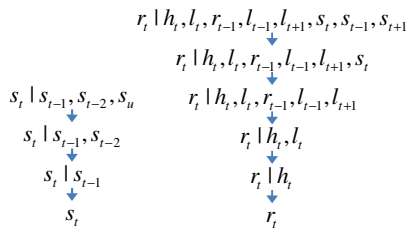


Figure 3: Backoff graphs for the semantic decoding and realisation models.

$$P(\mathbf{R}|\mathbf{S}^*) = \prod_{t=1}^{L} P(r_t|r_{t-1}, h_t, l_{t-1}, l_t, l_{t+1}, s^*_{t-1}, s^*_t, s^*_{t+1}) \quad (8)$$

Conditional probability distributions are represented as factored language models smoothed using Witten-Bell interpolated backoff smoothing (Bilmes and Kirchhoff, 2003), according to the backoff graphs in Fig. 3. Variables which are the furthest away in time are dropped first, and partial stack variables are dropped last as they are observed the most.

It is important to note that generating unseen semantic stacks requires all possible mandatory semantic stacks in the target domain to be predefined, in order for all stack unigrams to be assigned a smoothed non-zero probability.

3.2 High cardinality concept abstraction

While one should expect a trainable generator to learn multiple lexical realisations for low-cardinality semantic concepts, learning lexical realisations for high-cardinality database entries (e.g., proper names) would increase the number of model parameters prohibitively. We thus divide pre-terminal concepts in the semantic stacks into two types: (a) *enumerable* attributes whose values are associated with distinct semantic stacks in

our model (e.g., inform(pricerange(cheap))), and (b) *non-enumerable* attributes whose values are replaced by a generic symbol before training in both the utterance and the semantic stack (e.g., inform(name(X)). These symbolic values are then replaced in the surface realisation by the corresponding value in the input specification. A consequence is that our model can only learn synonymous lexical realisations for enumerable attributes.

4 Certainty-based active learning

A major issue with trainable NLG systems is the lack of availability of domain-specific data. It is therefore essential to produce NLG models that minimise the data annotation cost.

BAGEL supports the optimisation of the data collection process through active learning, in which the next semantic input to annotate is determined by the current model. The probabilistic nature of BAGEL allows the use of *certainty-based* active learning (Lewis and Catlett, 1994), by querying the k semantic inputs for which the model is the least certain about its output realisation. Given a finite semantic input space $\mathcal{I}$ representing all possible dialogue acts in our domain (i.e., the set of all sets of mandatory semantic stacks $\mathcal{S}_m$), BAGEL's active learning training process iterates over the following steps:

1. Generate an utterance for each semantic input $\mathcal{S}_m \in \mathcal{I}$ using the current model.[2]
2. Annotate the k semantic inputs $\{\mathcal{S}_m^1 ... \mathcal{S}_m^k\}$ yielding the lowest realisation probability, i.e. for $q \in (1..k)$

$$\mathcal{S}_m^q = \underset{\mathcal{S}_m \in \mathcal{I} \setminus \{\mathcal{S}_m^1 ... \mathcal{S}_m^{q-1}\}}{\operatorname{argmin}} (\max_{\mathbf{R}} P(\mathbf{R}|\mathcal{S}_m)) \quad (9)$$

   with $P(\mathbf{R}|\mathcal{S}_m)$ defined in (2).
3. Retrain the model with the additional k data points.

The number of utterances to be queried k should depend on the flexibility of the annotators and the time required for generating all possible utterances in the domain.

5 Experimental method

BAGEL's factored language models are trained using the SRILM toolkit (Stolcke, 2002), and decoding is performed using GMTK's junction tree inference algorithm (Bilmes and Zweig, 2002). Since each active learning iteration requires generating all training utterances in our domain, they are generated using a larger clique pruning threshold than the test utterances used for evaluation.

5.1 Corpus collection

We train BAGEL in the context of a dialogue system providing information about restaurants in Cambridge. The domain contains two dialogue act types: (a) inform: presenting information about a restaurant (see Table 1), and (b) reject: informing that the user's constraints cannot be met (e.g., 'There is no cheap restaurant in the centre'). Our domain contains 8 restaurant attributes: name, food, near, pricerange, postcode, phone, address, and area, out of which food, pricerange, and area are treated as enumerable.[3] Our input semantic space is approximated by the set of information presentation dialogue acts produced over 20,000 simulated dialogues between our statistical dialogue manager (Young et al., 2010) and an agenda-based user simulator (Schatzmann et al., 2007), which results in 202 unique dialogue acts after replacing non-enumerable values by a generic symbol. Each dialogue act contains an average of 4.48 mandatory semantic stacks.

As one of our objectives is to test whether BAGEL can learn from data provided by a large sample of untrained annotators, we collected a corpus of semantically-aligned utterances using Amazon's Mechanical Turk data collection service. A crucial aspect of data collection for NLG is to ensure that the annotators understand the meaning of the semantics to be conveyed. Annotators were first asked to provide an utterance matching an abstract description of the dialogue act, regardless of the order in which the constraints are presented (e.g., *Offer the venue Taj Mahal and provide the information type(restaurant), area(riverside), food(Indian), near(The Red Lion)*). The order of the constraints in the description was randomised to reduce the effect of priming. The annotators were then asked to align the attributes (e.g., *Indicate the region of the utterance related to the concept 'area'*), and the attribute values (e.g., *Indicate only the words related to the concept 'riverside'*). Two paraphrases were collected for each dialogue act in our domain, resulting in a total of 404 aligned ut-

[2] Sampling methods can be used if $\mathcal{I}$ is infinite or too large.

[3] With the exception of areas defined as proper nouns.

r_t	s_t	h_t	l_t
<s>	START	START	START
The Rice Boat	inform(name(X))	X	inform(name)
is a	inform	inform	EMPTY
restaurant	inform(type(restaurant))	restaurant	inform(type)
in the	inform(area)	area	inform
riverside	inform(area(riverside))	riverside	inform(area)
area	inform(area)	area	inform
that	inform	inform	EMPTY
serves	inform(food)	food	inform
French	inform(food(French))	French	inform(food)
food	inform(food)	food	inform
</s>	END	END	END

Table 2: Example utterance annotation used to estimate the conditional probability distributions of the models in Figs. 1 and 2 (r_t=realisation phrase, s_t=semantic stack, h_t=stack head, l_t=stack tail).

terances produced by 42 native speakers of English. After manually checking and normalising the dataset,[4] the layered annotations were automatically mapped to phrase-level semantic stacks by splitting the utterance into phrases at annotation boundaries. Each annotated utterance is then converted into a sequence of symbols such as in Table 2, which are used to estimate the conditional probability distributions defined in (6) and (8). The resulting vocabulary consists of 52 distinct semantic stacks and 109 distinct realisation phrases, with an average of 8.35 phrases per utterance.

5.2 BLEU score evaluation

We first evaluate BAGEL using the BLEU automated metric (Papineni et al., 2002), which measures the word n-gram overlap between the generated utterances and the 2 reference paraphrases over a test corpus (with n up to 4). While BLEU suffers from known issues such as a bias towards statistical NLG systems (Reiter and Belz, 2009), it provides useful information when comparing similar systems. We evaluate BAGEL for different training set sizes, model dependencies, and active learning parameters. Our results are averaged over a 10-fold cross-validation over distinct dialogue acts, i.e. dialogue acts used for testing are *not seen* at training time,[5] and all systems are tested on the same folds. The training and test sets respectively contain an average of 181 and 21 distinct dialogue acts, and each dialogue act is associated with two paraphrases, resulting in 362 training utterances.

[4]The normalisation process took around 4 person-hour for 404 utterances.

[5]We do not evaluate performance on dialogue acts used for training, as the training examples can trivially be used as generation templates.

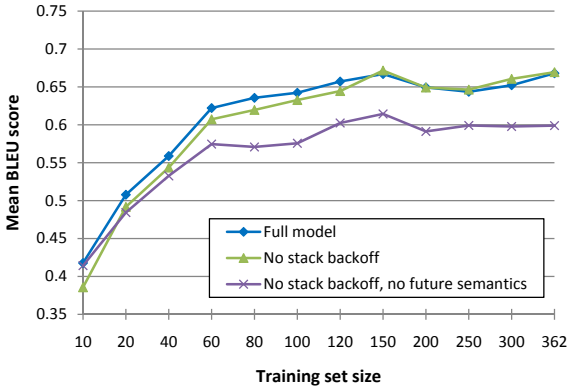


Figure 4: BLEU score averaged over a 10-fold cross-validation for different training set sizes and network topologies, using random sampling.

Results: Fig. 4 shows that adding a dependency on the future semantic stack improves performances for all training set sizes, despite the added model complexity. Backing off to partial stacks also improves performance, but only for sparse training sets.

Fig. 5 compares the full model trained using random sampling in Fig. 4 with the same model trained using certainty-based active learning, for different values of k. As our dataset only contains two paraphrases per dialogue act, the same dialogue act can only be queried twice during the active learning procedure. A consequence is that the training set used for active learning converges towards the randomly sampled set as its size increases. Results show that increasing the training set one utterance at a time using active learning ($k = 1$) significantly outperforms random sampling when using 40, 80, and 100 utterances ($p < .05$, two-tailed). Increasing the number of utterances to be queried at each iteration to $k = 10$ results in a smaller performance increase. A possi-

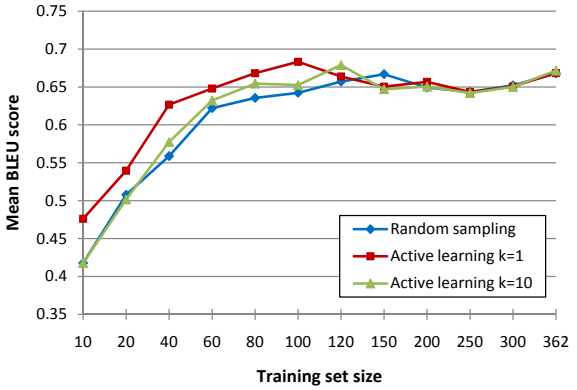


Figure 5: BLEU score averaged over a 10-fold cross-validation for different numbers of queries per iteration, using the full model with the query selection criterion (9).

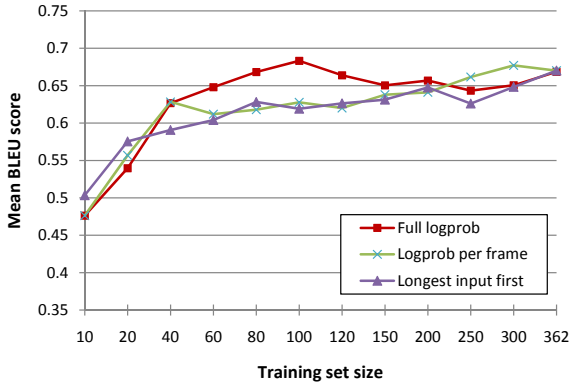


Figure 6: BLEU score averaged over a 10-fold cross-validation for different query selection criteria, using the full model with $k = 1$.

ble explanation is that the model is likely to assign low probabilities to similar inputs, thus any value above $k = 1$ might result in redundant queries within an iteration.

As the length of the semantic stack sequence is not known before decoding, the active learning selection criterion presented in (9) is biased towards longer utterances, which tend to have a lower probability. However, Fig. 6 shows that normalising the log probability by the number of semantic stacks does not improve overall learning performance. Although a possible explanation is that longer inputs tend to contain more information to learn from, Fig. 6 shows that a baseline selecting the largest remaining semantic input at each iteration performs worse than the active learning scheme for training sets above 20 utterances. The full log probability selection criterion defined in (9) is therefore used throughout the rest of the paper (with $k = 1$).

5.3 Human evaluation

While automated metrics provide useful information for comparing different systems, human feedback is needed to assess (a) the quality of BAGEL's outputs, and (b) whether training models using active learning has a significant impact on user perceptions. We evaluate BAGEL through a large-scale subjective rating experiment using Amazon's Mechanical Turk service.

For each dialogue act in our domain, participants are presented with a 'gold standard' human utterance from our dataset, which they must compare with utterances generated by models trained with and without active learning on a set of 20, 40, 100, and 362 utterances (full training set), as well as with the second human utterance in our dataset. See example utterances in Table 3. The judges are then asked to evaluate the *informativeness* and *naturalness* of each of the 8 utterances on a 5 point likert-scale. Naturalness is defined as whether the utterance could have been produced by a human, and informativeness is defined as whether it contains all the information in the gold standard utterance. Each utterance is taken from the test folds of the cross-validation experiment presented in Section 5.2, i.e. the models are trained on up to 90% of the data and the training set does not contain the dialogue act being tested.

Results: Figs. 7 and 8 compare the naturalness and informativeness scores of each system averaged over all 202 dialogue acts. A paired t-test shows that models trained on 40 utterances or less produce utterances that are rated significantly lower than human utterances for both naturalness and informativeness ($p < .05$, two-tailed). However, models trained on 100 utterances or more do not perform significantly worse than human utterances for both dimensions, with a mean difference below .10 over 202 comparisons. Given the large sample size, this result suggests that BAGEL can successfully learn our domain using a fraction of our initial dataset.

As far as the learning method is concerned, a paired t-test shows that models trained on 20 and 40 utterances using active learning significantly outperform models trained using random sampling, for both dimensions ($p < .05$). The largest increase is observed using 20 utterances, i.e. the naturalness increases by .49 and the informativeness by .37. When training on 100 utterances, the effect of active learning becomes insignificant. In-

Input	inform(name(the Fountain) near(the Arts Picture House) area(centre) pricerange(cheap))
Human	There is an inexpensive restaurant called the Fountain in the centre of town near the Arts Picture House
Rand-20	The Fountain is a restaurant near the Arts Picture House located in the city centre cheap price range
Rand-40	The Fountain is a restaurant in the cheap city centre area near the Arts Picture House
AL-20	The Fountain is a restaurant near the Arts Picture House in the city centre cheap
AL-40	The Fountain is an affordable restaurant near the Arts Picture House in the city centre
Full set	The Fountain is a cheap restaurant in the city centre near the Arts Picture House
Input	reject(area(Barnwell) near(Saint Mary's Church))
Human	I am sorry but I know of no venues near Saint Mary's Church in the Barnwell area
Full set	I am sorry but there are no venues near Saint Mary's Church in the Barnwell area
Input	inform(name(the Swan)area(Castle Hill) pricerange(expensive))
Human	The Swan is a restaurant in Castle Hill if you are seeking something expensive
Full set	The Swan is an expensive restaurant in the Castle Hill area
Input	inform(name(Browns) area(centre) near(the Crowne Plaza) near(El Shaddai) pricerange(cheap))
Human	Browns is an affordable restaurant located near the Crowne Plaza and El Shaddai in the centre of the city
Full set	Browns is a cheap restaurant in the city centre near the Crowne Plaza and El Shaddai

Table 3: Example utterances for different input dialogue acts and system configurations. *AL-20* = active learning with 20 utterances, *Rand* = random sampling.

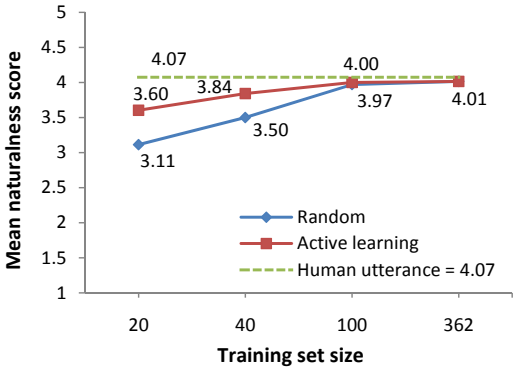


Figure 7: Naturalness mean opinion scores for different training set sizes, using random sampling and active learning. Differences for training set sizes of 20 and 40 are all significant ($p < .05$).

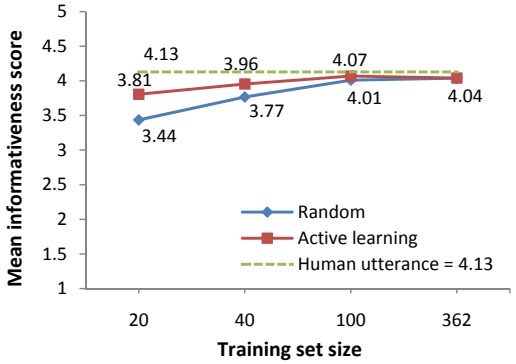


Figure 8: Informativeness mean opinion scores for different training set sizes, using random sampling and active learning. Differences for training set sizes of 20 and 40 are all significant ($p < .05$).

terestingly, while models trained on 100 utterances outperform models trained on 40 utterances using random sampling ($p < .05$), they do not significantly outperform models trained on 40 utterances using active learning ($p = .15$ for naturalness and $p = .41$ for informativeness). These results suggest that certainty-based active learning is beneficial for training a generator from a limited amount of data given the domain size.

Looking back at the results presented in Section 5.2, we find that the BLEU score correlates with a Pearson correlation coefficient of .42 with the mean naturalness score and .35 with the mean informativeness score, over all folds of all systems tested ($n = 70$, $p < .01$). This is lower than previous correlations reported by Reiter and Belz (2009) in the shipping forecast domain with non-expert judges ($r = .80$), possibly because our domain is larger and more open to subjectivity.

6 Related work

While most previous work on trainable NLG relies on a handcrafted component (see Section 1), recent research has started exploring fully data-driven NLG models.

Factored language models have recently been used for surface realisation within the OpenCCG framework (White et al., 2007; Espinosa et al., 2008). More generally, chart generators for different grammatical formalisms have been trained from syntactic treebanks (White et al., 2007; Nakanishi et al., 2005), as well as from semantically-annotated treebanks (Varges and Mellish, 2001). However, a major difference with our approach is that BAGEL uses domain-specific data to generate a surface form directly from semantic concepts, without any syntactic annotation (see Section 7 for further discussion).

This work is strongly related to Wong and Mooney's WASP^{-1} generation system (2007), which combines a language model with an inverted synchronous CFG parsing model, effectively casting the generation task as a translation problem from a meaning representation to natural language. WASP^{-1} relies on GIZA++ to align utterances with derivations of the meaning representation (Och and Ney, 2003). Although early experiments showed that GIZA++ did not perform well on our data—possibly because of the coarse granularity of our semantic representation—future work should evaluate the generalisation performance of synchronous CFGs in a dialogue system domain.

Although we do not know of any work on active learning for NLG, previous work has used active learning for semantic parsing and information extraction (Thompson et al., 1999; Tang et al., 2002), spoken language understanding (Tur et al., 2003), speech recognition (Hakkani-Tür et al., 2002), word alignment (Sassano, 2002), and more recently for statistical machine translation (Bloodgood and Callison-Burch, 2010). While certainty-based methods have been widely used, future work should investigate the performance of *committee-based* active learning for NLG, in which examples are selected based on the level of disagreement between models trained on subsets of the data (Freund et al., 1997).

7 Discussion and conclusion

This paper presents and evaluates BAGEL, a statistical language generator that can be trained entirely from data, with no handcrafting required beyond the semantic annotation. All the required subtasks—i.e. content ordering, aggregation, lexical selection and realisation—are learned from data using a unified model. To train BAGEL in a dialogue system domain, we propose a stack-based semantic representation at the phrase level, which is expressive enough to generate natural utterances from *unseen* inputs, yet simple enough for data to be collected from 42 untrained annotators with a minimal normalisation step. A human evaluation over 202 dialogue acts does not show any difference in naturalness and informativeness between BAGEL's outputs and human utterances. Additionally, we find that the data collection process can be optimised using active learning, resulting in a significant increase in performance when training data is limited, according to ratings from 18 human judges.[6] These results suggest that the proposed framework can largely reduce the development time of NLG systems.

While this paper only evaluates the most likely realisation given a dialogue act, we believe that BAGEL's probabilistic nature and generalisation capabilities are well suited to model the linguistic variation resulting from the diversity of annotators. Our first objective is thus to evaluate the quality of BAGEL's n-best outputs, and test whether sampling from the output distribution can improve naturalness and user satisfaction within a dialogue.

Our results suggest that explicitly modelling syntax is not necessary for our domain, possibly because of the lack of syntactic complexity compared with formal written language. Nevertheless, future work should investigate whether syntactic information can improve performance in more complex domains. For example, the realisation phrase can easily be conditioned on syntactic constructs governing that phrase, and the recursive nature of syntax can be modelled by keeping track of the depth of the current embedded clause. While syntactic information can be included with no human effort by using syntactic parsers, their robustness to dialogue system utterances must first be evaluated.

Finally, recent years have seen HMM-based synthesis models become competitive with unit selection methods (Tokuda et al., 2000). Our long term objective is to take advantage of those advances to jointly optimise the language generation and the speech synthesis process, by combining both components into a unified probabilistic concept-to-speech generation model.

References

S. Bangalore and O. Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 42–48, 2000.

A. Belz. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455, 2008.

J. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of HLT-NAACL, short papers*, 2003.

J. Bilmes and G. Zweig. The Graphical Models ToolKit: An open source software system for speech and time-series processing. In *Proceedings of ICASSP*, 2002.

[6] The full training corpus and the generated utterances used for evaluation are available at `http://mi.eng.cam.ac.uk/~farm2/bagel`.

M. Bloodgood and C. Callison-Burch. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2010.

D. Espinosa, M. White, and D. Mehay. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

Y. Freund, H. S. Seung, E.Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

D. Hakkani-Tür, G. Riccardi, and A. Gorin. Active learning for automatic speech recognition. In *Proceedings of ICASSP*, 2002.

Y. He and S. Young. Semantic processing using the Hidden Vector State model. *Computer Speech & Language*, 19 (1):85–106, 2005.

A. Isard, C. Brockmann, and J. Oberlander. Individuality and alignment in generated dialogues. In *Proceedings of the 4th International Natural Language Generation Conference (INLG)*, pages 22–29, 2006.

I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 704–710, 1998.

F. Lefèvre. A DBN-based multi-level stochastic spoken language understanding system. In *Proceedings of the IEEE Workshop on Spoken Language Technology (SLT)*, 2006.

D. D. Lewis and J. Catlett. Heterogeneous uncertainty ampling for supervised learning. In *Proceedings of ICML*, 1994.

F. Mairesse and M. A. Walker. Trainable generation of Big-Five personality styles through data-driven parameter estimation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

H. Nakanishi, Y. Miyao, , and J. Tsujii. Probabilistic methods for disambiguation of an HPSG-based chart generator. In *Proceedings of the IWPT*, 2005.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

D. S. Paiva and R. Evans. Empirically-based control of natural language generation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 58–65, 2005.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

L. R. Rabiner. Tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

E. Reiter and A. Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 25: 529–558, 2009.

V. Rieser and O. Lemon. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Proceedings of the Annual Meeting of the European Chapter of the ACL (EACL)*, 2009.

M. Sassano. An empirical study of active learning with support vector machines for japanese word segmentation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of HLT-NAACL, short papers*, pages 149–152, 2007.

A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, 2002.

M. Tang, X. Luo, and S. Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

C. Thompson, M. E. Califf, and R. J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of ICML*, 1999.

B. Thomson and S. Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

Y. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of ICASSP*, 2000.

G. Tur, R. E. Schapire, and D. Hakkani-Tür. Active learning for spoken language understanding. In *Proceedings of ICASSP*, 2003.

S. Varges and C. Mellish. Instance-based natural language generation. In *Proceedings of the Annual Meeting of the North American Chapter of the ACL (NAACL)*, 2001.

M. A. Walker, O. Rambow, and M. Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*, 16(3-4), 2002.

M. White, R. Rajkumar, and S. Martin. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation*, 2007.

Y. W. Wong and R. Mooney. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of HLT-NAACL*, 2007.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The Hidden Information State model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174, 2010.

Plot Induction and Evolutionary Search for Story Generation

Neil McIntyre and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, UK
n.d.mcintyre@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we develop a story generator that leverages knowledge inherent in corpora without requiring extensive manual involvement. A key feature in our approach is the reliance on a story planner which we acquire automatically by recording events, their participants, and their precedence relationships in a training corpus. Contrary to previous work our system does not follow a generate-and-rank architecture. Instead, we employ evolutionary search techniques to explore the space of possible stories which we argue are well suited to the story generation task. Experiments on generating simple children's stories show that our system outperforms previous data-driven approaches.

1 Introduction

Computer story generation has met with fascination since the early days of artificial intelligence. Indeed, over the years, several generators have been developed capable of creating stories that resemble human output. To name only a few, TALE-SPIN (Meehan, 1977) generates stories through problem solving, MINSTREL (Turner, 1992) relies on an episodic memory scheme, essentially a repository of previous hand-coded stories, to solve the problems in the current story, and MAKEBELIEVE (Liu and Singh, 2002) uses commonsense knowledge to generate short stories from an initial seed story (supplied by the user). A large body of more recent work views story generation as a form of agent-based planning (Swartjes and Theune, 2008; Pizzi et al., 2007). The agents act as characters with a list of goals. They form plans of action and try to fulfill them. Interesting stories emerge as plans interact and cause failures and possible replanning.

The broader appeal of computational story generation lies in its application potential. Examples include the entertainment industry and the development of tools that produce large numbers of plots automatically that might provide inspiration to professional screen writers (Agudo et al., 2004); rendering video games more interesting by allowing the plot to adapt dynamically to the players' actions (Barros and Musse, 2007); and assisting teachers to create or personalize stories for their students (Riedl and Young, 2004).

A major stumbling block for the widespread use of computational story generators is their reliance on expensive, manually created resources. A typical story generator will make use of a knowledge base for providing detailed domain-specific information about the characters and objects involved in the story and their relations. It will also have a story planner that specifies how these characters interact, what their goals are and how their actions result in different story plots. Finally, a sentence planner (coupled with a surface realizer) will render an abstract story specification into natural language text. Traditionally, most of this knowledge is created by hand, and the effort must be repeated for new domains, new characters and plot elements.

Fortunately, recent work in natural language processing has taken significant steps towards developing algorithms that learn some of this knowledge *automatically* from natural language corpora. Chambers and Jurafsky (2009, 2008) propose an unsupervised method for learning *narrative schemas*, chains of events whose arguments are filled with participant semantic roles defined over words. An example schema is {X arrest, X charge, X raid, X seize, X confiscate, X detain, X deport}, where X stands for the argument types {*police, agent, authority, government*}. Their approach relies on the intuition that in a coherent text events that are about the same participants are

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1562–1572,
Uppsala, Sweden, 11-16 July 2010.

likely to be part of the same story or narrative. Their model extracts narrative chains, essentially events that share argument slots and merges them into schemas. The latter could be used to construct or enrich the knowledge base of a story generator.

In McIntyre and Lapata (2009) we presented a story generator that leverages knowledge inherent in corpora without requiring extensive manual involvement. The generator operates over predicate-argument and predicate-predicate co-occurrence tuples gathered from training data. These are used to produce a large set of candidate stories which are subsequently ranked based on their interestingness and coherence. The approach is unusual in that it does not involve an explicit story planning component. Stories are created stochastically by selecting entities and the events they are most frequently attested with.

In this work we develop a story generator that is also data-driven but crucially relies on a story planner for creating meaningful stories. Inspired by Chambers and Jurafsky (2009) we acquire story plots automatically by recording events, their participants, and their precedence relationships as attested in a training corpus. Entities give rise to different potential plots which in turn generate multiple stories. Contrary to our previous work (McIntyre and Lapata, 2009), we do not follow a generate-and-rank architecture. Instead, we search the space of possible stories using Genetic Algorithms (GAs) which we argue are advantageous in the story generation setting, as they can search large fitness landscapes while greatly reducing the risk of getting stuck in local optima. By virtue of exploring the search space more broadly, we are able to generate creative stories without an explicit interest scoring module.

In the remainder of this paper we give a brief overview of the system described in McIntyre and Lapata (2009) and discuss previous applications of GAs in natural language generation (Section 2). Next, we detail our approach, specifically how plots are created and used in conjunction with genetic search (Sections 3 and 4). Finally, we present our experimental results (Sections 6 and 7) and conclude the paper with discussion of future work.

2 Related Work

Our work builds on and extends the story generator developed in McIntyre and Lapata (2009). The system creates simple children's stories in an interactive context: the user supplies the topic of the story and its desired length (number of sentences). The generator creates a story following a pipeline architecture typical of natural language generation systems (Reiter and Dale, 2000) consisting of content selection, sentence planning, and surface realization.

The content of a story is determined by consulting a data-driven knowledge base that records the entities (i.e., nouns) appearing in a corpus and the actions they perform. These are encoded as dependency relations (e.g., *subj-verb*, *verb-obj*). In order to promote between-sentence coherence the generator also make use of an *action graph* that contains action-role pairs and the likelihood of transitioning from one to another. The sentence planner aggregates together entities and their actions into a sentence using phrase structure rules. Finally, surface realization is performed by interfacing RealPro (Lavoie and Rambow, 1997) with a language model. The system searches for the best story overall as well as the best sentences that can be generated from the knowledge base. Unlikely stories are pruned using beam search. In addition, stories are reranked using two scoring functions based on coherence and interest. These are learnt from training data, i.e., stories labeled with numeric values for interest and coherence.

Evolutionary search techniques have been previously employed in natural language generation, especially in the context of document planning. Structuring a set of facts into a coherent text is effectively a search problem that may lead to combinatorial explosion for large domains. Mellish et al. (1998) (and subsequently Karamanis and Manurung 2002) advocate genetic algorithms as an alternative to exhaustively searching for the optimal ordering of descriptions of museum artefacts. Rather than requiring a global optimum to be found, the genetic algorithm selects an order (based on coherence) that is good enough for people to understand. Cheng and Mellish (2000) focus on the interaction of aggregation and text planning and use genetic algorithms to search for the best aggregated document that satisfies coherence constraints.

The application of genetic algorithms to story generation is novel to our knowledge. Our work also departs from McIntyre and Lapata (2009) in two important ways. Firstly, our generator does not rely on a knowledge base of seemingly unrelated entities and relations. Rather, we employ

a document planner to create and structure a plot for a story. The planner is built automatically from a training corpus and creates plots dynamically depending on the protagonists of the story. Secondly, our search procedure is simpler and more global; instead of searching for the best story twice (i.e., by first finding the *n*-best stories and then subsequently reranking them based on coherence and interest), our genetic algorithm explores the space of possible stories once.

3 Plot Generation

Following previous work (e.g., Shim and Kim 2002; McIntyre and Lapata 2009) we assume that the user supplies a sentence (e.g., *the princess loves the prince*) from which the system creates a story. Each entity in this sentence (e.g., *princess*, *prince*) is associated with its own *narrative schema*, a set of key events and actors co-occurring with it in the training corpus. Our narrative schemas differ slightly from Chambers and Jurafsky (2009). They acquire schematic representations of situations akin to FrameNet (Fillmore et al., 2003): schemas consists of semantically similar predicates and the entities evoked by them. In our setting, every entity has its own schema, and predicates associated with it are ordered. Plots are generated by merging the entity-specific narrative schemas which subsequently serve as the input to the genetic algorithm. In the following we describe how the narrative schemas are extracted and plots merged, and then discuss our evolutionary search procedure.

Entity-based Schema Extraction Before we can generate a plot for a story we must have an idea of the actions associated with the entities in the story, the order in which these actions are performed and also which other entities can participate. This information is stored in a directed graph which we explain below. Our algorithm processes each document at a time, it operates over dependency structures and assumes that entity mentions have been resolved. In our experiments we used Rasp (Briscoe and Carroll, 2002), a broad coverage dependency parser, and the OpenNLP[1] coreference resolution engine.[2] However, any dependency parser or coreference tool could serve our purpose. We also assume that the actions associated with a given entity are ordered and that linear order corresponds to temporal order. This is a gross simplification as it is well known that temporal relationships between events are not limited to precedence, they may overlap, occur simultaneously, or be temporally unrelated. We could have obtained a more accurate ordering using a temporal classifier (see Chambers and Jurafsky 2008), however we leave this to future work.

For each entity e in the corpus we build a directed graph $G = (V, E)$ whose nodes V denote predicate argument relationships, and edges E represent transitions from node V_i to node V_j. As an example of our schema construction process, consider a very small corpus consisting of the two documents shown in Figure 1. The schema for *princess* after processing the first document is given on the left hand side. Each node in this graph corresponds to an action attested with *princess* (we also record who performs it and where or how). Nodes are themselves dependency trees (see Figure 4a), but are linearized in the figure for the sake of brevity. Edges in the graph indicate ordering and are weighted using the mutual information metric proposed in Lin (1998) (the weights are omitted from the example).[3] The first sentence in the text gives rise to the first node in the graph, the second sentence to the second node, and so on. Note that the third sentence is not present in the graph as it is not about the *princess*.

When processing the second document, we simply expand this graph. Before inserting a new node, we check if it can be merged with an already existing one. Nodes are merged only if they have the same verb and similar arguments, with the focal entity (i.e., *princess*) appearing in the same argument slot. In our example, the nodes "prince marry princess in castle" and "prince marry princess in temple" can be merged as they contain the same verb and number of similar arguments. The nodes "princess have influence" and "princess have baby" cannot be merged as *influence* and *baby* are semantically unrelated. We compute argument similarity using WordNet (Fellbaum, 1998) and the measure proposed by Wu and Palmer (1994) which is based on path length. We merge nodes with related arguments only if their similarity exceeds a threshold (determined empirically).

[1] See `http://opennlp.sourceforge.net/`.

[2] The coreference resolution tool we employ is not error-free and on occasion will fail to resolve a pronoun. We map unresolved pronouns to the generic labels *person* or *object*.

[3] We use mutual information to identify event sequences strongly associated with the graph entity.

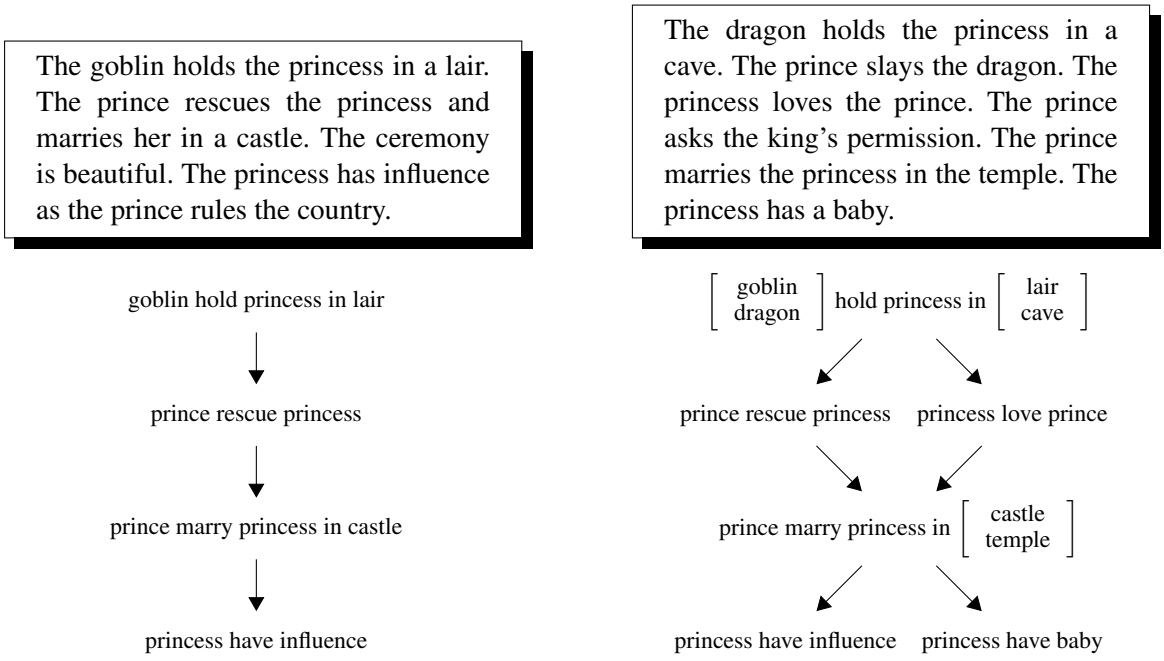


Figure 1: Example of schema construction for the entity *princess*

The schema construction algorithm terminates when graphs like the ones shown in Figure 1 (right hand side) have been created for all entities in the corpus.

Building a Story Plot Our generator takes an input sentence and uses it to instantiate several plots. We achieve this by merging the schemas associated with the entities in the sentence into a plot graph. As an example, consider again the sentence *the princess loves the prince* which requires combing the schemas representing *prince* and *princess* shown in Figures 2 and 1 (right hand side), respectively. Again, we look for nodes that can be merged based on the identity of the actions involved and the (WordNet) similarity of their arguments. However, we disallow the merging of nodes with focal entities appearing in the same argument slot (e.g., "[prince, princess] cries").

Once the plot graph is created, a depth first search starting from the node corresponding to the input sentence, finds all paths with length matching the desired story length (cycles are disallowed). Assuming we wish to generate a story consisting of three sentences, the graph in Figure 3 would create four plots. These are (princess love prince, prince marry princess in [castle, temple], princess have influence), (princess love prince, prince marry princess in [castle, temple], princess have baby), (princess love prince, prince marry princess in [castle, temple], prince rule country), and (princess love prince, prince ask king's permission prince marry princess in [castle, temple]). Each of these plots represents two different stories one with *castle* and one with *temple* in it.

Sentence Planning The sentence planner is interleaved with the story planner and influences the final structure of each sentence in the story. To avoid generating short sentences — note that nodes in the plot graph consist of a single action and would otherwise correspond to a sentence with a single clause — we combine pairs of nodes within the same graph by looking at intrasentential verb-verb co-occurrences in the training corpus. For example, the nodes (prince have problem, prince keep secret) could become the sentence *the prince has a problem keeping a secret*. We leave it up to the sentence planner to decide how the two actions should be combined.[4] The sentence planner will also insert adverbs and adjectives, using co-occurrence likelihoods acquired from the training corpus. It is essentially a phrase structure grammar compiled from the lexical resources made available by Korhonen and Briscoe (2006) and Grishman et al. (1994). The grammar rules act as templates for combining clauses and filling argument slots.

[4] We only turn an action into a subclause if its subject entity is same as that of the previous action.

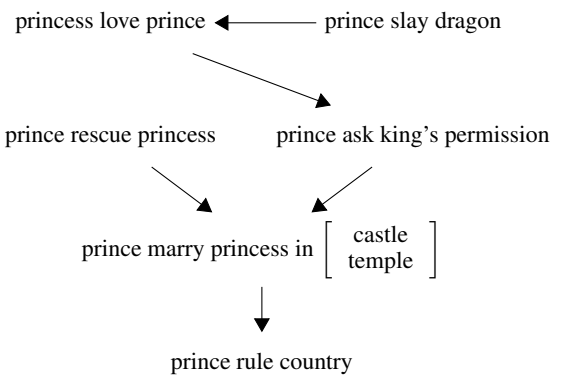


Figure 2: Narrative schema for the entity *prince*.

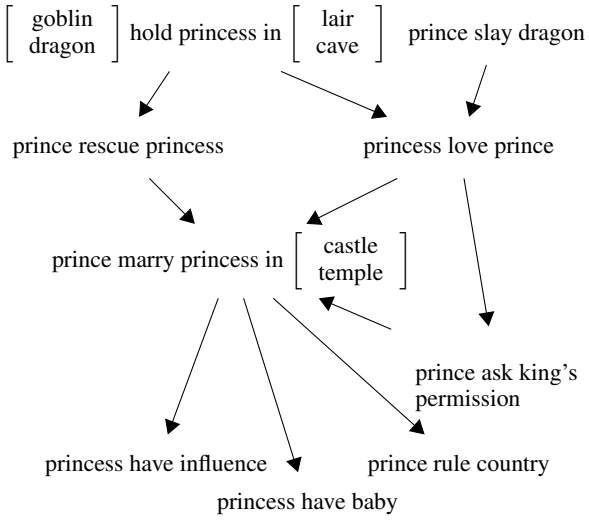


Figure 3: Plot graph for the input sentence *the princess loves the prince*.

4 Genetic Algorithms

The example shown in Figure 3 is a simplified version of a plot graph. The latter would normally contain hundreds of nodes and give rise to thousands of stories once lexical variables have been expanded. Searching the story space is a difficult optimization problem, that must satisfy several constraints: the story should be of a certain length, overall coherent, creative, display some form of event progression, and generally make sense. We argue that evolutionary search is appealing here, as it can find global optimal solutions in a more efficient way than traditional optimization methods.

In this study we employ genetic algorithms (GAs) a well-known search technique for finding approximate (or exact) solutions to optimization problems. The basic idea behind GAs is based on "natural selection" and the Darwinian principle of the survival of the fittest (Mitchell, 1998). An initial population is randomly created containing a predefined number of individuals (or solutions), each represented by a genetic string (e.g., a population of chromosomes). Each individual is evaluated according to an objective function (also called a fitness function). A number of individuals are then chosen as parents from the population according to their fitness, and undergo *crossover* (also called recombination) and *mutation* in order to develop the new population. Offspring with better fitness are then inserted into the population, replacing the inferior individuals in the previous generation.

The algorithm thus identifies the individuals with the optimizing fitness values, and those with lower fitness will naturally get discarded from the population. This cycle is repeated for a given number of generations, or stopped when the solution obtained is considered optimal. This process leads to the evolution of a population in which the individuals are more and more suited to their environment, just as natural adaptation. We describe below how we developed a genetic algorithm for our story generation problem.

Initial Population Rather than start with a random population, we seed the initial population with story plots generated from our plot graph. For an input sentence, we generate all possible plots. The latter are then randomly sampled until a population of the desired size is created. Contrary to McIntyre and Lapata (2009), we initialize the search with complete stories, rather than generate one sentence at a time. The genetic algorithm will thus avoid the pitfall of selecting early on a solution that will later prove detrimental.

Crossover Each plot is represented as an ordered graph of dependency trees (corresponding to sentences). We have decided to use crossover of a single point between two selected parents. The children will therefore contain sentences up to the crossover point of the first parent and sentences after that point of the second. Figure 4a shows two parents (prince rescue princess, prince marry princess in castle, princess have baby) and (prince rescue princess, prince love princess, princess kiss prince) and how two new plots are created by swapping their last nodes.

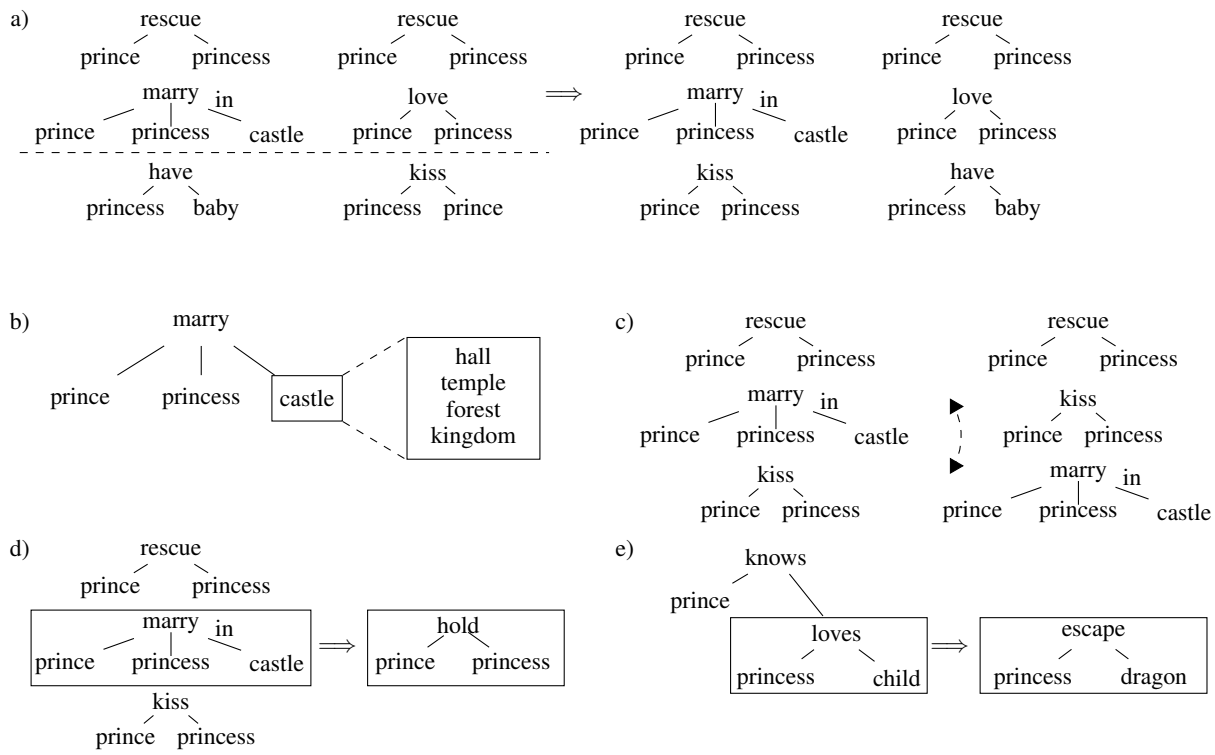


Figure 4: Example of genetic algorithm operators as they are applied to plot structures: a) crossover of two plots on a single point, indicated by the dashed line, resulting in two children which are a recombination of the parents; b) mutation of a lexical node, *church* can be replaced from a list of semantically related candidates; c) sentences can be switched under mutation to create a potentially more coherent structure; d) if the matrix verb undergoes mutation then, a random sentence is generated to replace it; e) if the verb chosen for mutation is the head of a subclause, then a random subclause replaces it.

Mutation Mutation can occur on any verb, noun, adverb, or adjective in the plot. If a noun, adverb or adjective is chosen to undergo mutation, then we simply substitute it with a new lexical item that is sufficiently similar (see Figure 4b for an example). Verbs, however, have structural importance in the stories and we cannot simply replace them without taking account of their arguments. If a matrix verb is chosen to undergo mutation, then a new random sentence is generated to replace the entire sentence (see Figure 4d). If it is a subclause, then it is replaced with a randomly generated clause, headed by a verb that has been seen in the corpus to co-occur with the matrix verb (Figure 4e). The sentence planner selects and fills template trees for generating random clauses. Mutation may also change the order of any two nodes in the graph in the hope that this will increase the story's coherence or create some element of surprise (see Figure 4c).

Selection To choose the plots for the next generation, we used fitness proportional selection (also know as roulette-wheel selection, Goldberg 1989) which chooses candidates randomly but with a bias towards those with a larger proportion of the population's combined fitness. We do not want to always select the fittest candidates as there may be valid partial solutions held within less fit members of the population. However, we did employ some elitism by allowing the top 1% of solutions to be copied straight from one generation to the next. Note that our candidates may also represent invalid solutions. For instance, through crossover it is possible to create a plot in which all or some nodes are identical. If any such candidates are identified, they are assigned a low fitness, without however being eliminated from the population as some could be used to create fitter solutions.

In a traditional GA, the fitness function deals with one optimization objective. It is possible to optimize several objectives either using a vot-

ing model or more sophisticated methods such as Pareto ranking (Goldberg, 1989). Following previous work (Mellish et al., 1998) we used a single fitness function that scored candidates based on their coherence. Our function was learned from training data using the Entity Grid document representation proposed in Barzilay and Lapata (2007). An entity grid is a two-dimensional array in which columns correspond to entities and rows to sentences. Each cell indicates whether an entity appears in a given sentence or not and whether it is a subject, object or neither. For training, this representation is converted into a feature vector of entity transition sequences and a model is learnt from examples of coherent and incoherent stories. The latter can be easily created by permuting the sentences of coherent stories (assuming that the original story is more coherent than its permutations).

In addition to coherence, in McIntyre and Lapata (2009) we used a scoring function based on interest which we approximated with lexical and syntactic features such as the number of noun/verb tokens/types, the number of subjects/objects, the number of letters, word familiarity, imagery, and so on. An interest-based scoring function made sense in our previous setup as a means of selecting unusual stories. However, in the context of genetic search such a function seems redundant as interesting stories emerge naturally through the operations of crossover and mutation.

5 Surface Realization

Once the final generation of the population has been reached, the fittest story is selected for surface realization. The realizer takes each sentence in the story and reformulates it into input compatible with the RealPro (Lavoie and Rambow, 1997) text generation engine. Realpro creates several variants of the same story differing in the choice of determiners, number (singular or plural), and prepositions. A language model is then used to select the most probable realization (Knight and Hatzivassiloglou, 1995). Ideally, the realizer should also select an appropriate tense for the sentence. However, we make the simplifying assumption that all sentences are in the present tense.

6 Experimental Setup

In this section we present our experimental set-up for assessing the performance of our story generator. We give details on our training corpus, system, parameters (such as the population size for the GA search), the baselines used for comparison, and explain how our system output was evaluated.

Corpus The generator was trained on the same corpus used in McIntyre and Lapata (2009), 437 stories from the Andrew Lang fairy tales collection.[5] The average story length is 125.18 sentences. The corpus contains 15,789 word tokens. Following McIntyre and Lapata, we discarded tokens that did not appear in the Children's Printed Word Database[6], a database of printed word frequencies as read by children aged between five and nine. From this corpus we extracted narrative schemas for 667 entities in total. We disregarded any graph that contained less than 10 nodes as too small. The graphs had on average 61.04 nodes, with an average clustering rate[7] of 0.027 which indicates that they are substantially connected.

Parameter Setting Considerable latitude is available when selecting parameters for the GA. These involve the population size, crossover, and mutation rates. To evaluate which setting was best, we asked two human evaluators to rate (on a 1–5 scale) stories produced with a population size ranging from 1,000 to 10,000, crossover rate of 0.1 to 0.6 and mutation rate of 0.001 to 0.1. For each run of the system a limit was set to 5,000 generations. The human ratings revealed that the best stories were produced for a population size of 10,000, a crossover rate of 0.1% and a mutation rate of 0.1%. Compared to previous work (e.g., Karamanis and Manurung 2002) our crossover rate may seem low and the mutation rate high. However, it makes intuitively sense, as high crossover may lead to incoherence by disrupting canonical action sequences found in the plots. On the other hand, a higher mutation will raise the likelihood of a lexical item being swapped for another and may improve overall coherence and interest. The fitness function was trained on 200 documents from the fairy tales collection using Joachims's (2002) SVM^{light} package and entity transition sequences of length 2. The realizer was interfaced with a trigram language model trained on the British National Corpus with the SRI toolkit.

[5] Available from `http://homepages.inf.ed.ac.uk/s0233364/McIntyreLapata09/`.

[6] `http://www.essex.ac.uk/psychology/cpwd/`

[7] Clustering rate (or transitivity) is the number of triangles in the graph — sets of three vertices each of which is connected to each of the others.

Evaluation We compared the stories generated by the GA against those produced by the rank-based system described in McIntyre and Lapata (2009) and a system that creates stories from the plot graph, without any stochastic search. Since plot graphs are weighted, we can simply select the graph with the highest weight. After expanding all lexical variables, the chosen plot graph will give rise to different stories (e.g., *castle* or *temple* in the example above). We select the story ranked highest according to our coherence function. In addition, we included a baseline which randomly selects sentences from the training corpus provided they contain either of the story protagonists (i.e., entities in the input sentence). Sentence length was limited to 12 words or less as this was on average the length of the sentences generated by our GA system.

Each system created stories for 12 input sentences, resulting in 48 (4×12) stories for evaluation. The sentences used commonly occurring entities in the fairy tales corpus (e.g., *The child watches the bird*, *The emperor rules the kingdom.*, *The wizard casts the spell.*). The stories were split into three sets containing four stories from each system but with only one story from each input sentence. All stories had the same length, namely five sentences. Human judges were presented with one of the three sets and asked to rate the stories on a scale of 1 to 5 for fluency (was the sentence grammatical?), coherence (does the story make sense overall?) and interest (how interesting is the story?). The stories were presented in random order and participants were told that all of them were generated by a computer program. They were instructed to rate more favorably interesting stories, stories that were comprehensible and overall grammatical. The study was conducted over the Internet using WebExp (Keller et al., 2009) and was completed by 56 volunteers, all self reported native English speakers.

7 Results

Our results are summarized in Table 1 which lists the average human ratings for the four systems. We performed an Analysis of Variance (ANOVA) to examine the effect of system type on the story generation task. Statistical tests were carried out on the mean of the ratings shown in Table 1 for fluency, coherence, and interest.

In terms of interest, the GA-based system is significantly better than the Rank-based, Plot-based and Random ones (using a Post-hoc Tukey test, $\alpha < 0.05$). With regard to fluency, the Rank-based system is significantly worse than the rest ($\alpha < 0.01$). Interestingly, the sentences generated by the GA and Plot-based systems are as fluent as those created by humans. Recall that the Random system, simply selects sentences from the training corpus. Finally, the GA system is significantly more coherent than the Rank-based and Random systems ($\alpha < 0.05$), but not the Plot-based one. This is not surprising, the GA and Plot-based systems rely on similar plots to create a coherent story. The performance of the Random system is also inferior as it does not have any explicit coherence enforcing mechanism. The Rank-based system is perceived overall worse. As this system is also the least fluent, we conjecture that participants are influenced in their coherence judgments by the grammaticality of the stories.

System	Fluency	Coherence	Interest
GA-based	3.09	2.48	2.36
Plot-based	3.03	2.36	2.14^{*}
Rank-based	1.96^{**}	1.65^{*}	1.85^{*}
Random	3.10	2.23^{*}	2.20^{*}

Table 1: Human evaluation results: mean story ratings for four story generators; $^{*}: p < 0.05$, $^{**}: p < 0.01$, significantly different from GA-based system.

Overall our results indicate that an explicit story planner improves the quality of the generated stories, especially when coupled with a search mechanism that advantageously explores the search space. It is worth noting that the Plot-based system is relatively simple, however the explicit use of a story plot, seems to make up for the lack of sophisticated search and more elaborate linguistic information. Example stories generated by the four systems are shown in Table 2 for the input sentences *The emperor rules the kingdom* and *The child watches the bird*.

Possible extensions and improvements to the current work are many and varied. Firstly, we could improve the quality of our plot graphs by taking temporal knowledge into account and making use of knowledge bases such as WordNet and ConceptNet (Liu and Davenport, 2004), a freely available commonsense knowledge base. Secondly, our fitness function optimizes one ob-

PlotGA	The emperor rules the kingdom. The kingdom holds on to the emperor. The emperor rides out of the kingdom. The kingdom speaks out against the emperor. The emperor lies.	The child watches the bird. The bird weeps for the child. The child begs the bird to listen.The bird dresses up the child. The child grows up.
Plot	The emperor rules the kingdom. The emperor takes over. The emperor goes on to feel for the kingdom. Possibly the emperor sleeps. The emperor steals.	The child watches the bird. The bird comes to eat away at the child. The child does thoroughly. The bird sees the child. The child sits down.
Rank	The emperor rules the kingdom. The kingdom lives from the reign to the emperor. The emperor feels that the brothers tempt a beauty into the game. The kingdom saves the life from crumbling the earth into the bird. The kingdom forces the whip into wiping the tears on the towel.	The child watches the bird. The bird lives from the reign to the child. The child thanks the victory for blessing the thought. The child loves to hate the sun with the thought. The child hopes to delay the duty from the happiness.
Random	Exclaimed the emperor when Petru had put his question. In the meantime, mind you take good care of our kingdom. At first the emperor felt rather distressed. The dinner of an emperor! Thus they arrived at the court of the emperor.	They cried, “what a beautiful child!” “No, that I cannot do, my child” he said at last. “What is the matter, dear child?” “You wicked child,” cried the Witch. Well, I will watch till the bird comes.

Table 2: Stories generated by a system that uses plots and genetic search (PlotGA), a system that uses only plots (Plot), McIntyre and Lapata (2009)’s rank-based system (Rank) and a system that randomly pastes together sentences from the training corpus (Random).

jective, namely coherence. In the future we plan to explore multiple objectives, such as whether the story is verbose, readable (using existing readability metrics), has two many or two few protagonists, and so on.

Thirdly, our stories would benefit from some explicit modeling of discourse structure. Although the plot graph captures the progression of the actions in a story, we would also like to know where in the story these actions are likely to occur—some tend to appear in the beginning and others in the end. Such information would allow us to structure the stories better and render them more natural sounding. For example, an improvement would be the inclusion of proper endings, as the stories are currently cut off at an arbitrary point when the desired maximum length is reached.

Finally, the fluency of the stories would benefit from generating referring expressions, multiple tense forms, indirect speech, aggregation and generally more elaborate syntactic structure.

References

Agudo, Belén Diáz, Pablo Gervás, and Frederico Peinado. 2004. A case based reasoning approach to story plot generation. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer, Madrid, Spain, pages 142–156.

Barros, Leandro Motta and Soraia Raupp Musse. 2007. Planning algorithms for interactive storytelling. In *Computers in Entertainment (CIE)*, Association for Computing Machinery (ACM), volume 5.

Barzilay, Regina and Mirella Lapata. 2007. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Briscoe, E. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*. Las Palmas, Gran Canaria, pages 1499–1504.

Chambers, Nathanael and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Columbus, Ohio, pages 789–797.

Chambers, Nathanael and Dan Jurafsky. 2009.

Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore, pages 602–610.

Cheng, Hua and Chris Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of the 1st International Conference on Natural Language Generation*. Mitzpe Ramon, Israel, pages 186–193.

Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, Cambridge, Massachusetts.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography* 16:235–250.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts.

Grishman, Ralph, Catherine Macleod, and Adam Meyers. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th COLING*. Kyoto, Japan, pages 268–272.

Joachims, Thorsten. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. Edmonton, Alberta, pages 133–142.

Karamanis, Nikiforos and Hisar Maruli Manurung. 2002. Stochastic text structuring using the principle of continuity. In *Proceedings of the 2nd International Natural Language Generation Conference (INLG'02)*. pages 81–88.

Keller, Frank, Subahshini Gunasekharan, Neil Mayo, and Martin Corley. 2009. Timing accuracy of web experiments: A case study using the WebExp software package. *Behavior Research Methods* 41(1):1–12.

Knight, Kevin and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*. Cambridge, Massachusetts, pages 252–260.

Korhonen, Y. Krymolowski, A. and E.J. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of the 5th LREC*. Genova, Italy.

Lavoie, Benoit and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing*. Washington, D.C., pages 265–268.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistic*. Montreal, Quebec, pages 768–774.

Liu, Hugo and Glorianna Davenport. 2004. ConceptNet: a practical commonsense reasoning toolkit. *BT Technology Journal* 22(4):211–226.

Liu, Hugo and Push Singh. 2002. Using commonsense reasoning to generate stories. In *Proceedings of the 18th National Conference on Artificial Intelligence*. Edmonton, Alberta, pages 957–958.

McIntyre, Neil and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Singapore, pages 217–225.

Meehan, James. 1977. An interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*. Cambridge, Massachusetts, pages 91–98.

Mellish, Chris, Alisdair Knott, Jon Oberlander, and Mick O'Donnell. 1998. Experiments using stochastic search for text planning. In *Proceedings of the 9th International Conference on Natural Language Generation*. New Brunswick, New Jersey, pages 98–107.

Mitchell, Melanie. 1998. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, Massachusetts.

Pizzi, David, Fred Charles, Jean-Luc Lugrin, and Marc Cavazza. 2007. Interactive storytelling with literary feelings. In *Proceedings of the 2nd International Conference on Affective Computing and Intelligent Interaction*. Lisbon, Portugal, pages 630–641.

Reiter, E and R Dale. 2000. *Building Natural-Language Generation Systems*. Cambridge University Press, Cambridge, UK.

Riedl, Mark O. and R. Michael Young. 2004. A planning approach to story generation and history education. In *Proceedings of the 3rd International Conference on Narrative and Interactive Learning Environments*. Edinburgh, UK, pages 41–48.

Shim, Yunju and Minkoo Kim. 2002. Automatic short story generator based on autonomous agents. In *Proceedings of the 5th Pacific Rim International Workshop on Multi Agents*. Tokyo, pages 151–162.

Swartjes, I.M.T. and M. Theune. 2008. The virtual storyteller: story generation by simulation. In *Proceedings of the 20th Belgian-Netherlands Conference on Artificial Intelligence, BNAIC 2008*. Enschede, the Netherlands, pages 257–264.

Turner, Scott R. 1992. *Ministrel: A Computer Model of Creativity and Sotrytelling*. University of California, Los Angeles, California.

Wu, Zhibiao and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, New Mexico, pages 133–138.

Automated planning for situated natural language generation

Konstantina Garoufi and **Alexander Koller**
Cluster of Excellence "Multimodal Computing and Interaction"
Saarland University, Saarbrücken, Germany
{garoufi,koller}@mmci.uni-saarland.de

Abstract

We present a natural language generation approach which models, exploits, and manipulates the non-linguistic context in situated communication, using techniques from AI planning. We show how to generate instructions which deliberately guide the hearer to a location that is convenient for the generation of simple referring expressions, and how to generate referring expressions with context-dependent adjectives. We implement and evaluate our approach in the framework of the Challenge on Generating Instructions in Virtual Environments, finding that it performs well even under the constraints of real-time generation.

1 Introduction

The problem of situated natural language generation (NLG)—i.e., of generating natural language in the context of a physical (or virtual) environment—has received increasing attention in the past few years. On the one hand, this is because it is the foundation of various emerging applications, including human-robot interaction and mobile navigation systems, and is the focus of a current evaluation effort, the Challenges on Generating Instructions in Virtual Environments (GIVE; (Koller et al., 2010b)). On the other hand, situated generation comes with interesting theoretical challenges: Compared to the generation of pure text, the interpretation of expressions in situated communication is sensitive to the non-linguistic context, and this context can change as easily as the user can move around in the environment.

One interesting aspect of situated communication from an NLG perspective is that this non-linguistic context can be manipulated by the speaker. Consider the following segment of discourse between an instruction giver (IG) and an instruction follower (IF), which is adapted from the SCARE corpus (Stoia et al., 2008):

(1) IG: *Walk forward and then turn right.*
IF: (walks and turns)
IG: *OK. Now hit the button in the middle.*

In this example, the IG plans to refer to an object (here, a button); and in order to do so, gives a navigation instruction to guide the IF to a convenient location at which she can then use a simple referring expression (RE). That is, there is an interaction between navigation instructions (intended to manipulate the non-linguistic context in a certain way) and referring expressions (which exploit the non-linguistic context). Although such subdialogues are common in SCARE, we are not aware of any previous research that can generate them in a computationally feasible manner.

This paper presents an approach to generation which is able to model the effect of an utterance on the non-linguistic context, and to intentionally generate utterances such as the above as part of a process of referring to objects. Our approach builds upon the CRISP generation system (Koller and Stone, 2007), which translates generation problems into planning problems and solves these with an AI planner. We extend the CRISP planning operators with the perlocutionary effects that uttering a particular word has on the physical environment if it is understood correctly; more specifically, on the position and orientation of the hearer. This allows the planner to predict the non-linguistic context in which a later part of the utterance will be interpreted, and therefore to search for contexts that allow the use of simple REs. As a result, the work of referring to an object gets distributed over multiple utterances of low cognitive load rather than a single complex noun phrase.

A second contribution of our paper is the generation of REs involving context-dependent adjectives: A button can be described as "the left blue

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1573–1582,
Uppsala, Sweden, 11-16 July 2010.

button" even if there is a red button to its left. We model adjectives whose interpretation depends on the nominal phrases they modify, as well as on the non-linguistic context, by keeping track of the distractors that remain after uttering a series of modifiers. Thus, unlike most other RE generation approaches, we are not restricted to building an RE by simply intersecting lexically specified sets representing the extensions of different attributes, but can correctly generate expressions whose meaning depends on the context in a number of ways. In this way we are able to refer to objects earlier and more flexibly.

We implement and evaluate our approach in the context of a GIVE NLG system, by using the GIVE-1 software infrastructure and a GIVE-1 evaluation world. This shows that our system generates an instruction-giving discourse as in (1) in about a second. It outperforms a mostly non-situated baseline significantly, and compares well against a second baseline based on one of the top-performing systems of the GIVE-1 Challenge. Next to the practical usefulness this evaluation establishes, we argue that our approach to jointly modeling the grammatical and physical effects of a communicative action can also inform new models of the pragmatics of speech acts.

Plan of the paper. We discuss related work in Section 2, and review the CRISP system, on which our work is based, in Section 3. We then show in Section 4 how we extend CRISP to generate navigation-and-reference discourses as in (1), and add context-dependent adjectives in Section 5. We evaluate our system in Section 6; Section 7 concludes and points to future work.

2 Related work

The research reported here can be seen in the wider context of approaches to generating referring expressions. Since the foundational work of Dale and Reiter (1995), there has been a considerable amount of literature on this topic. Our work departs from the mainstream in two ways. First, it exploits the situated communicative setting to deliberately modify the context in which an RE is generated. Second, unlike most other RE generation systems, we allow the contribution of a modifier to an RE to depend both on the context and on the rest of the RE.

We are aware of only one earlier study on generation of REs with focus on interleaving navigation and referring (Stoia et al., 2006). In this machine learning approach, Stoia et al. train classifiers that signal when the context conditions (e.g. visibility of target and distractors) are appropriate for the generation of an RE. This method can be then used as part of a content selection component of an NLG system. Such a component, however, can only inform a system on whether to choose navigation over RE generation at a given point of the discourse, and is not able to help it decide what kind of navigational instructions to generate so that subsequent REs become simple.

To our knowledge, the only previous research on generating REs with context-dependent modifiers is van Deemter's (2006) algorithm for generating vague adjectives. Unlike van Deemter, we integrate the RE generation process tightly with the syntactic realization, which allows us to generate REs with more than one context-dependent modifier and model the effect of their linear order on the meaning of the phrase. In modeling the context, we focus on the non-linguistic context and the influence of each of the RE's words; this is in contrast to previous research on context-sensitive generation of REs, which mainly focused on the discourse context (Krahmer and Theune, 2002). Our interpretation of context-dependent modifiers picks up ideas by Kamp and Partee (1995) and implements them in a practical system, while our method of ordering modifiers is linguistically informed by the class-based paradigm (e.g., Mitchell (2009)).

On the other hand, our work also stands in a tradition of NLG research that is based on AI planning. Early approaches (Perrault and Allen, 1980; Appelt, 1985) provided compelling intuitions for this connection, but were not computationally viable. The research we report here can be seen as combining Appelt's idea of using planning for sentence-level NLG with a computationally benign variant of Perrault et al.'s approach of modeling the intended perlocutionary effects of a speech act as the effects of a planning operator. Our work is linked to a growing body of very recent work that applies modern planning research to various problems in NLG (Steedman and Petrick, 2007; Brenner and Kruijff-Korbayová, 2008; Benotti, 2009). It is directly based on Koller and Stone's (2007) reimplementation of the SPUD generator (Stone et al., 2003) with planning. As far as we know, ours is the first system in the SPUD tradi-

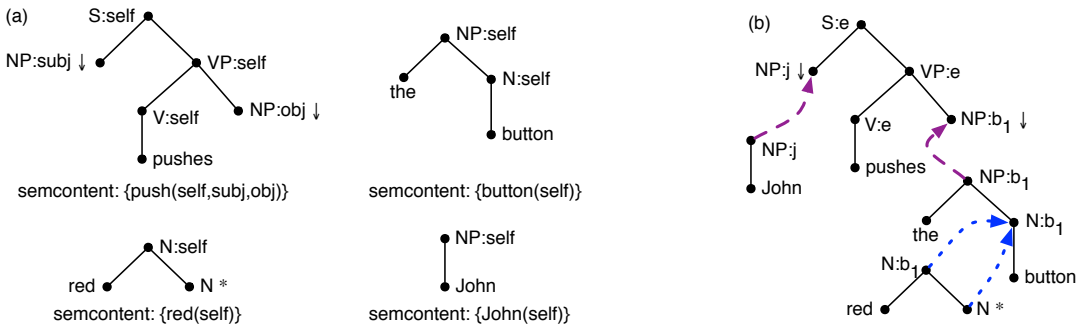


Figure 1: (a) An example grammar; (b) a derivation of "John pushes the red button" using (a).

tion that explicitly models the context change effects of an utterance.

While nothing in our work directly hinges on this, we implemented our approach in the context of an NLG system for the GIVE Challenge (Koller et al., 2010b), that is, as an instruction giving system for virtual worlds. This makes our system comparable with other approaches to instruction giving implemented in the GIVE framework.

3 Sentence generation as planning

Our work is based on the CRISP system (Koller and Stone, 2007), which encodes sentence generation with tree-adjoining grammars (TAG; (Joshi and Schabes, 1997)) as an AI planning problem and solves that using efficient planners. It then decodes the resulting plan into a TAG derivation, from which it can read off a sentence. In this section, we briefly recall how this works. For space reasons, we will present primarily examples instead of definitions.

3.1 TAG sentence generation

The CRISP generation problem (like that of SPUD (Stone et al., 2003)) assumes a lexicon of entries consisting of a TAG elementary tree annotated with semantic and pragmatic information. An example is shown in Fig. 1a. In addition to the elementary tree, each lexicon entry specifies its *semantic content* and possibly a *semantic requirement*, which can express certain presuppositions triggered by this entry. The nodes in the tree may be labeled with argument names such as *semantic roles*, which specify the participants in the relation expressed by the lexicon entry; in the example, every entry uses the semantic role self representing the event or individual itself, and the entry for "pushes" furthermore uses subj and obj for the subject and object argument, respectively. We combine here for simplicity the entries for "the" and "button" into "the button".

For generation, we assume as input a knowledge base and a communicative goal in addition to the grammar. The goal is to compute a derivation that expresses the communicative goal in a sentence that is grammatically correct and complete; whose meaning is justified by the knowledge base; and in which all REs can be resolved to unique individuals in the world by the hearer. Let's say, for example, that we have a knowledge base $\{\mathsf{push}(e, \mathsf{j}, b_1), \mathsf{John}(\mathsf{j}), \mathsf{button}(b_1), \mathsf{button}(b_2), \mathsf{red}(b_1)\}$. Then we can combine instances of the trees for "John", "pushes", and "the button" into a grammatically complete derivation. However, because both b_1 and b_2 satisfy the semantic content of "the button", we must adjoin "red" into the derivation to make the RE refer uniquely to b_1. The complete derivation is shown in Fig. 1b; we can read off the output sentence "John pushes the red button" from the leaves of the derived tree we build in this way.

3.2 TAG generation as planning

In the CRISP system, Koller and Stone (2007) show how this generation problem can be solved by converting it into a planning problem (Nau et al., 2004). The basic idea is to encode the partial derivation in the planning state, and to encode the action of adding each elementary tree in the planning operators. The encoding of our example as a planning problem is shown in Fig. 2.

In the example, we start with an initial state which contains the entire knowledge base, plus atoms $\mathsf{subst}(\mathrm{S}, \mathsf{root})$ and $\mathsf{ref}(\mathsf{root}, e)$ expressing that we want to generate a sentence about the event e. We can then apply the (instantiated) action $\textbf{pushes}(\mathsf{root}, \mathsf{n}_1, \mathsf{n}_2, \mathsf{n}_3, e, \mathsf{j}, b_1)$, which models the act of substituting the elementary tree for "pushes"

pushes$(u, u_1, u_2, u_n, x, x_1, x_2)$:
Precond: $\mathsf{subst}(\mathrm{S}, u), \mathsf{ref}(u, x), \mathsf{push}(x, x_1, x_2), \mathsf{current}(u_1), \mathsf{next}(u_1, u_2), \mathsf{next}(u_2, u_n)$
Effect: $\neg\mathsf{subst}(\mathrm{S}, u), \mathsf{subst}(\mathrm{NP}, u_1), \mathsf{subst}(\mathrm{NP}, u_2), \mathsf{ref}(u_1, x_1), \mathsf{ref}(u_2, x_2), \forall y.\mathsf{distractor}(u_1, y), \forall y.\mathsf{distractor}(u_2, y)$

John(u, x):
Precond: $\mathsf{subst}(\mathrm{NP}, u), \mathsf{ref}(u, x), \mathsf{John}(x)$
Effect: $\neg\mathsf{subst}(\mathrm{NP}, u), \forall y.\neg\mathsf{John}(y) \rightarrow \neg\mathsf{distractor}(u, y)$

the-button(u, x):
Precond: $\mathsf{subst}(\mathrm{NP}, u), \mathsf{ref}(u, x), \mathsf{button}(x)$
Effect: $\neg\mathsf{subst}(\mathrm{NP}, u), \mathsf{canadjoin}(\mathrm{N}, u), \forall y.\neg\mathsf{button}(y) \rightarrow \neg\mathsf{distractor}(u, y)$

red(u, x):
Precond: $\mathsf{canadjoin}(\mathrm{N}, u), \mathsf{ref}(u, x), \mathsf{red}(x)$
Effect: $\forall y.\neg\mathsf{red}(y) \rightarrow \neg\mathsf{distractor}(u, y)$

Figure 2: CRISP planning operators for the elementary trees in Fig. 1.

into the substitution node root: It can only be applied because root is an unfilled substitution node (precondition $\mathsf{subst}(\mathrm{S}, \mathsf{root})$), and its effect is to remove $\mathsf{subst}(\mathrm{S}, \mathsf{root})$ from the planning state while adding two new atoms $\mathsf{subst}(\mathrm{NP}, \mathsf{n}_1)$ and $\mathsf{subst}(\mathrm{NP}, \mathsf{n}_2)$ for the substitution nodes of the "pushes" tree. The planning state maintains information about which individual each node refers to in the ref atoms. The current and next atoms are needed to select unused names for newly introduced syntax nodes.[1] Finally, the action introduces a number of distractor atoms including $\mathsf{distractor}(\mathsf{n}_2, e)$ and $\mathsf{distractor}(\mathsf{n}_2, b_2)$, expressing that the RE at n_2 can still be misunderstood by the hearer as e or b_2.

In this new state, all subst and distractor atoms for n_1 can be eliminated with the action **John**$(\mathsf{n}_1, \mathsf{j})$. We can also apply the action **the-button**(n_2, b_1) to eliminate $\mathsf{subst}(\mathrm{NP}, \mathsf{n}_2)$ and $\mathsf{distractor}(\mathsf{n}_2, e)$, since e is not a button. However $\mathsf{distractor}(\mathsf{n}_2, b_2)$ remains. Now because the action **the-button** also introduced the atom $\mathsf{canadjoin}(\mathrm{N}, \mathsf{n}_2)$, we can remove the final distractor atom by applying **red**(n_2, b_1). This brings us into a goal state, and we are done. Goal states in CRISP planning problems are characterized by axioms such as $\forall A \forall u.\neg\mathsf{subst}(A, u)$ (encoding grammatical completeness) and $\forall u \forall x.\neg\mathsf{distractor}(u, x)$ (requiring unique reference).

[1] This is a different solution to the name-selection problem than in Koller and Stone (2007). It is simpler and improves computational efficiency.

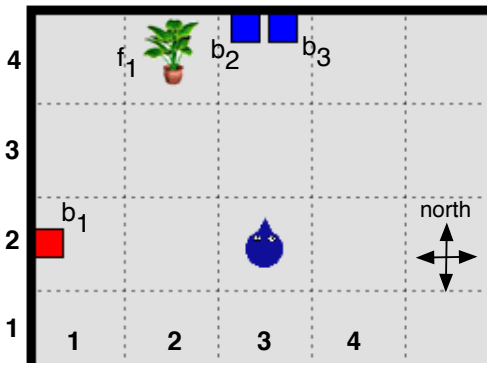


Figure 3: An example map for instruction giving.

3.3 Decoding the plan

An AI planner such as FF (Hoffmann and Nebel, 2001) can compute a plan for a planning problem that consists of the planning operators in Fig. 2 and a specification of the initial state and the goal. We can then decode this plan into the TAG derivation shown in Fig. 1b. The basic idea of this decoding step is that an action with a precondition $\mathsf{subst}(A, u)$ fills the substitution node u, while an action with a precondition $\mathsf{canadjoin}(A, u)$ adjoins into a node of category A in the elementary tree that was substituted into u. CRISP allows multiple trees to adjoin into the same node. In this case, the decoder executes the adjunctions in the order in which they occur in the plan.

4 Context manipulation

We are now ready to describe our NLG approach, SCRISP ("Situated CRISP"), which extends CRISP to take the non-linguistic context of the generated utterance into account, and deliberately manipulate it to simplify RE generation.

As a simplified version of our introductory instruction giving example (1), consider the map in Fig. 3. The instruction follower (IF), who is located on the map at position $\mathsf{pos}_{3,2}$ facing north, sees the scene from the first-person perspective as in Fig. 7. Now an instruction giver (IG) could instruct the IF to press the button b_1 in this scene by saying "push the button on the wall to your left". Interpreting this instruction is difficult for the IF because it requires her to either memorize the RE until she has turned to see the button, or to perform a mental rotation task to visualize b_1 internally. Alternatively, the IG can first instruct the IF to "turn left"; once the IF has done this, the IG can then simply say "now push the button in front

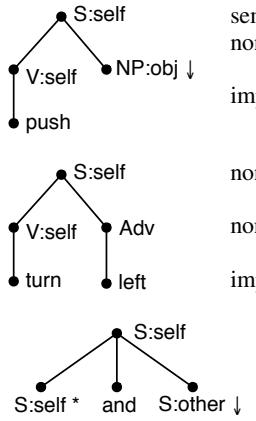


Figure 4: An example SCRISP lexicon.

turnleft(u, x, o_1, o_2):
Precond: subst(S, u), ref(u, x), player–ori(o_1), next–ori–left(o_1, o_2), ...
Effect: ¬subst(S, u), ¬player–ori(o_1), player–ori(o_2), to–do(turnleft), ...

push$(u, u_1, u_n, x, x_1, p, o)$:
Precond: subst(S, u), ref(u, x), player–pos(p), player–ori(o), visible(p, o, x_1), ...
Effect: ¬subst(S, u), subst(NP, u_1), ref(u_1, x_1), $\forall y.(y \neq x_1 \wedge \text{visible}(p, o, y) \rightarrow \text{distractor}(u_1, y))$, to–do(push($x_1$)), canadjoin(S, u), ...

and(u, u_1, u_n, e_1, e_2):
Precond: canadjoin(S, u), ref(u, e_1), ...
Effect: subst(S, u_1), ref(u_1, e_2), ...

Figure 5: SCRISP planning operators for the lexicon in Fig. 4.

of you". This lowers the cognitive load on the IF, and presumably improves the rate of correctly interpreted REs.

SCRISP is capable of deliberately generating such context-changing navigation instructions. The key idea of our approach is to extend the CRISP planning operators with preconditions and effects that describe the (simulated) physical environment: A "turn left" action, for example, modifies the IF's orientation in space and changes the set of visible objects; a "push" operator can then pick up this changed set and restrict the distractors of the forthcoming RE it introduces (i.e. "the button") to only objects that are visible in the changed context. We also extend CRISP to generate imperative rather than declarative sentences.

4.1 Situated CRISP

We define a lexicon for SCRISP to be a CRISP lexicon in which every lexicon entry may also describe *non-linguistic conditions*, *non-linguistic effects* and *imperative effects*. Each of these is a set of atoms over constants, semantic roles, and possibly some free variables. Non-linguistic conditions specify what must be true in the world so a particular instance of a lexicon entry can be uttered felicitously; non-linguistic effects specify what changes uttering the word brings about in the world; and imperative effects contribute to the IF's "to-do list" (Portner, 2007) by adding the properties they denote.

A small lexicon for our example is shown in Fig. 4. This lexicon specifies that saying "push X" puts pushing X on the IF's to-do list, and carries the presupposition that X must be visible from the location where "push X" is uttered; this reflects our simplifying assumption that the IG can only refer to objects that are currently visible. Similarly, "turn left" puts turning left on the IF's agenda. In addition, the lexicon entry for "turn left" specifies that, under the assumption that the IF understands and follows the instruction, they will turn 90 degrees to the left after hearing it. The planning operators are written in a way that assumes that the intended (perlocutionary) effects of an utterance actually come true. This assumption is crucial in connecting the non-linguistic effects of one SCRISP action to the non-linguistic preconditions of another, and generalizes to a scalable model of planning perlocutionary acts. We discuss this in more detail in Koller et al. (2010a).

We then translate a SCRISP generation problem into a planning problem. In addition to what CRISP does, we translate all non-linguistic conditions into preconditions and all non-linguistic effects into effects of the planning operator, adding any free variables to the operator's parameters. An imperative effect P is translated into an effect to–do(P). The operators for the example lexicon of Fig. 4 are shown in Fig. 5. Finally, we add information about the situated environment to the initial state, and specify the planning goal by adding to–do(P) atoms for each atom P that is to be placed on the IF's agenda.

4.2 An example

Now let's look at how this generates the appropriate instructions for our example scene of Fig. 3. We encode the state of the world as depicted in the map in an initial state which contains, among others, the atoms player–pos($\text{pos}_{3,2}$), player–ori(north), next–ori–left(north, west),

visible($\mathsf{pos}_{3,2}$, west, b_1), etc.[2] We want the IF to press b_1, so we add to–do(push(b_1)) to the goal.

We can start by applying the action **turnleft**(root, e, north, west) to the initial state. Next to the ordinary grammatical effects from CRISP, this action makes player–ori(west) true. The new state does not contain any subst atoms, but we can continue the sentence by adjoining "and", i.e. by applying the action **and**(root, n_1, n_2, e, e_1). This produces a new atom subst(S, e_1), which satisfies one precondition of **push**(n_1, n_2, n_3, e_1, b_1, $\mathsf{pos}_{3,2}$, west). Because **turnleft** changed the player orientation, the visible precondition of **push** is now satisfied too (unlike in the initial state, in which b_1 was not visible). Applying the action **push** now introduces the need to substitute a noun phrase for the object, which we can eliminate with an application of **the-button**(n_2, b_1) as in Subsection 3.2.

Since there are no other visible buttons from $\mathsf{pos}_{3,2}$ facing west, there are no remaining distractor atoms at this point, and a goal state has been reached. Together, this four-step plan decodes into the sentence "turn left and push the button". The final state contains the atoms to–do(push(b_1)) and to–do(turnleft), indicating that an IF that understands and accepts this instruction also accepts these two commitments into their to-do list.

5 Generating context-dependent adjectives

Now consider if we wanted to instruct the IF to press b_2 in Fig. 3 instead of b_1, say with the instruction "push the left button". This is still challenging, because (like most other approaches to RE generation) CRISP interprets adjectives by simply intersecting all their extensions. In the case of "left", the most reasonable way to do this would be to interpret it as "leftmost among all visible objects"; but this is f_1 in the example, and so there is no distinguishing RE for b_2.

In truth, spatial adjectives like "left" and "upper" depend on the context in two different ways. On the one hand, they are interpreted with respect to the current spatio-visual context, in that what is on the left depends on the current position and orientation of the hearer. On the other hand, they also depend on the meaning of the phrase they modify: "the left button" is not necessarily both a button and further to the left than all other objects, it is only the leftmost object among the buttons.

We will now show how to extend SCRISP so it can generate REs that use such context-dependent adjectives.

left(u, x):
  Precond: $\forall y.\neg$(distractor(u, y) $\wedge$ left–of(y, x)),
    canadjoin(N, u), ref(u, x)
  Effect: $\forall y.$(left–of(x, y) $\rightarrow \neg$distractor(u, y)),
    premod–index(u, 2), ...

red(u, x):
  Precond: red(x), canadjoin(N, u), ref(u, x),
    $\neg$premod–index(u, 2)
  Effect: $\forall y.(\neg$red(y) $\rightarrow \neg$distractor(u, y)),
    premod–index(u, 1), ...

Figure 6: SCRISP operators for context-dependent and context-independent adjectives.

5.1 Context-dependence of adjectives in SCRISP

As a planning-based approach to NLG, SCRISP is not limited to simply intersecting sets of potential referents that only depend on the attributes that contribute to an RE: Distractors are removed by applying operators which may have context-sensitive conditions depending on the referent and the distractors that are still left.

Our encoding of context-dependent adjectives as planning operators is shown in Fig. 6. We only show the operators here for lack of space; they can of course be computed automatically from lexicon entries. In addition to the ordinary CRISP preconditions, the **left** operator has a precondition requiring that no current distractor for the RE u is to the left of x, capturing a presupposition of the adjective. Its effect is that everything that is to the right of x is no longer a distractor for u. Notice that we allow that there may still be distractors after **left** has been applied (above or below x); we only require unique reference in the goal state. (Ignore the premod–index part of the effect for now; we will get to that in a moment.)

Let's say that we are computing a plan for referring to b_2 in the example map of Fig. 3, starting with **push**(root, n_1, n_2, e, b_2, $\mathsf{pos}_{3,1}$, north) and **the-button**(n_1, b_2). The state after these two actions is not a goal state, because it still contains the atom distractor(n_1, b_3) (the plant f_1 was removed as a distractor by the action **the-button**).

[2] In a more complex situation, it may be infeasible to exhaustively model visibility in this way. This could be fixed by connecting the planner to an external spatial reasoner (Dornhege et al., 2009).

Now assume that we have modeled the spatial relations between all objects in the initial state in left–of and above atoms; in particular, we have left–of(b_2, b_3). Then the action instance **left**(n_1, b_2) is applicable in this state, as there is no other object that is still a distractor in this state and that is to the left of b_2. Applying **left** removes distractor(n_1, b_3) from the state. Thus we have reached a goal state; the complete plan decodes to the sentence "push the left button".

This system is sensitive to the order in which operators for context-dependent adjectives are applied. To generate the RE "the upper left button", for instance, we first apply the **left** action and then the **upper** action, and therefore **upper** only needs to remove distractors in the leftmost position. On the other hand, the RE "the left upper button" corresponds to first applying **upper** and then **left**. These action sequences succeed in removing all distractors for different context states, which is consistent with the difference in meaning between the two REs.

Furthermore, notice that the adjective operators themselves do not interact directly with the encoding of the context in atoms like visible and player–pos, just like the noun operators in Section 4 didn't. The REs to which the adjectives and nouns contribute are introduced by verb operators; it is these verb operators that inspect the current context and initialize the distractor set for the new RE appropriately. This makes the correctness of the generated sentence independent of the order in which noun and adjective operators occur in the plan. We only need to ensure that the verbs are ordered correctly, and the workload of modeling interactions with the non-linguistic context is limited to a single place in the encoding.

5.2 Adjective word order

One final challenge that arises in our system is to generate the adjectives in the correct order, which on top of semantically valid must be linguistically acceptable. In particular, it is known that some types of adjectives are limited with respect to the word order in which they can occur in a noun phrase. For instance, "large foreign financial firms" sounds perfectly acceptable, but "? foreign large financial firms" sounds odd (Shaw and Hatzivassiloglou, 1999). In our setting, some adjective orders are forbidden because only one order produces a correct and distinguishing description of the target referent (cf. "upper left" vs. "left upper" example above). However, there are also other constraints at work: "? the red left button" is rather odd even when it is a semantically correct description, whereas "the left red button" is fine.

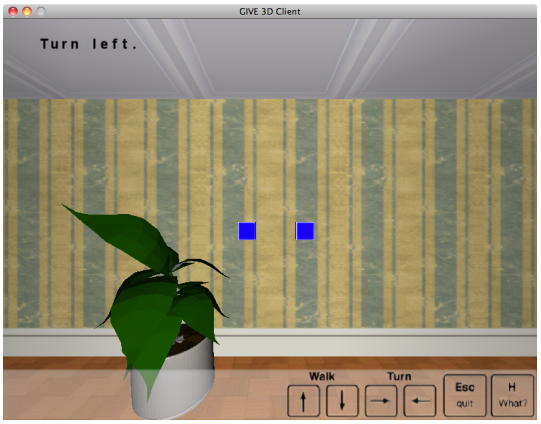


Figure 7: The IF's view of the scene in Fig. 3, as rendered by the GIVE client.

To ensure that SCRISP chooses to generate these adjectives correctly, we follow a class-based approach to the premodifier ordering problem (Mitchell, 2009). In our lexicon we assign adjectives denoting spatial relations ("left") to one class and adjectives denoting color ("red") to another; then we require that spatial adjectives must always precede color adjectives. We enforce this by keeping track of the current *premodifier index* of the RE in atoms of the form premod–index. Any newly generated RE node starts off with a premodifier index of zero; adjoining an adjective of a certain class then raises this number to the index for that class. As the operators in Fig. 6 illustrate, color adjectives such as "red" have index one and can only be used while the index is not higher; once an adjective from a higher class (such as "left", of a class with index two) is used, the premod–index precondition of the "red" operator will fail. For this reason, we can generate a plan for "the left red button", but not for "? the red left button", as desired.

6 Evaluation

To establish the quality of the generated instructions, we implemented SCRISP as part of a generation system in the GIVE-1 framework, and evaluated it against two baselines. GIVE-1 was the First Challenge on Generating Instructions in Virtual Environments, which was completed in 2009

SCRISP	*1. Turn right and move one step.* *2. Push* ***the right red button****.*
Baseline A	*1. Press* ***the right red button on the wall to your right****.*
Baseline B	*1. Turn right.* *2. Walk forward 3 steps.* *3. Turn right.* *4. Walk forward 1 step.* *5. Turn left.* *6. Good! Now press* ***the left button****.*

Table 1: Example system instructions generated in the same scene. REs for the target are typeset in boldface.

(Koller et al., 2010b). In this challenge, systems must generate real-time instructions that help users perform a task in a treasure-hunt virtual environment such as the one shown in Fig. 7.

We conducted our evaluation in World 2 from GIVE-1, which was deliberately designed to be challenging for RE generation. The world consists of one room filled with several objects and buttons, most of which cannot be distinguished by simple descriptions. Moreover, some of those may activate an alarm and cause the player to lose the game. The player's moves and turns are discrete and the NLG system has complete and accurate real-time information about the state of the world. Instructions that each of the three systems under comparison generated in an example scene of the evaluation world are presented in Table 1.

The evaluation took place online via the Amazon Mechanical Turk, where we collected 25 games for each system. We focus on four measures of evaluation: success rates for solving the task and resolving the generated REs, average task completion time (in seconds) for successful games, and average distance (in steps) between the IF and the referent at the time when the RE was generated. As in the challenge, the task is considered as solved if the player has correctly been led through manipulating all target objects required to discover and collect the treasure; in World 2, the minimum number of such targets is eight. An RE is successfully resolved if it results in the manipulation of the referent, whereas manipulation of an alarm-triggering distractor ends the game unsuccessfully.

6.1 The SCRISP system

Our system receives as input a plan for what the IF should do to solve the task, and successively takes object-manipulating actions as the communicative goals for SCRISP. Then, for each of the communicative goals, it generates instructions using SCRISP, segments them into navigation and action parts, and presents these to the user as separate instructions sequentially (see Table 1).

For each instruction, SCRISP thus draws from a knowledge base of about 1500 facts and a grammar of about 30 lexicon entries. We use the FF planner (Hoffmann and Nebel, 2001; Koller and Hoffmann, 2010) to solve the planning problems. The maximum planning time for any instruction is 1.03 seconds on a 3.06 GHz Intel Core 2 Duo CPU. So although our planning-based system tackles a very difficult search problem, FF is very good at solving it—fast enough to generate instructions in real time.

	success		**RE**	
	rate	**time**	**success**	**distance**
SCRISP	69%	306	71%	2.49
Baseline A	16%**	230	49%**	1.97*
Baseline B	84%	288	81%*	2.00*

Table 2: Evaluation results. Differences to SCRISP are significant at $*p < .05$, $**p < .005$ (Pearson's chi-square test for system success rates; unpaired two-sample t-test for the rest).

6.2 Comparison with Baseline A

Baseline A is a very basic system designed to simulate the performance of a classical RE generation module which does not attempt to manipulate the visual context. We hand-coded a correct distinguishing RE for each target button in the world; the only way in which Baseline A reacts to changes of the context is to describe on which wall the button is with respect to the user's current orientation (e.g. "Press the right red button *on the wall to your right*").

As Table 2 shows, our system guided 69% of users to complete the task successfully, compared to only 16% for Baseline A (difference is statistically significant at $p < .005$; Pearson's chi-square test). This is primarily because only 49% of the REs generated by Baseline A were successful. This comparison illustrates the importance of REs that minimize the cognitive load on the IF to avoid misunderstandings.

6.3 Comparison with Baseline B

Baseline B is a corrected and improved version of the "Austin" system (Chen and Karpov, 2009),

one of the best-performing systems of the GIVE-1 Challenge. Baseline B, like the original "Austin" system, issues navigation instructions by precomputing the shortest path from the IF's current location to the target, and generates REs using the description logic based algorithm of Areces et al. (2008). Unlike the original system, which inflexibly navigates the user all the way to the target, Baseline B starts off with navigation, and opportunistically instructs the IF to push a button once it has become visible and can be described by a distinguishing RE. We fixed bugs in the original implementation of the RE generation module, so that Baseline B generates only unambiguous REs. The module nonetheless naively treats all adjectives as intersective and is not sensitive to the context of their comparison set. Specifically, a button cannot be referred to as "the *right* red button" if it is not the rightmost of all visible objects—which explains the long chain of navigational instructions the system produced in Table 1.

We did not find any significant differences in the success rates or task completion times between this system and SCRISP, but the former achieved a higher RE success rate (see Table 2). However, a closer analysis shows that SCRISP was able to generate REs from significantly further away. This means that SCRISP's RE generator solves a harder problem, as it typically has to deal with more visible distractors. Furthermore, because of the increased distance, the system's execution monitoring strategies (e.g. for detection and repair of misunderstandings) become increasingly important, and this was not a focus of this work. In summary, then, we take the results to mean that SCRISP performs quite capably in comparison to a top-ranked GIVE-1 system.

7 Conclusion

In this paper, we have shown how situated instructions can be generated using AI planning. We exploited the planner's ability to model the perlocutionary effects of communicative actions for efficient generation. We showed how this made it possible to generate instructions that manipulate the non-linguistic context in convenient ways, and to generate correct REs with context-dependent adjectives.

We believe that this illustrates the power of a planning-based approach to NLG to flexibly model very different phenomena. An interesting topic for future work, for instance, is to expand our notion of context by taking visual and discourse salience into account when generating REs. In addition, we plan to experiment with assigning costs to planning operators in a metric planning problem (Hoffmann, 2002) in order to model the cognitive cost of an RE (Krahmer et al., 2003) and compute minimal-cost instruction sequences.

On a more theoretical level, the SCRISP actions model the physical effects of a correctly understood and grounded instruction directly as effects of the planning operator. This is computationally much less complex than classical speech act planning (Perrault and Allen, 1980), in which the intended physical effect comes at the end of a long chain of inferences. But our approach is also very optimistic in estimating the perlocutionary effects of an instruction, and must be complemented by an appropriate model of execution monitoring. What this means for a novel scalable approach to the pragmatics of speech acts (Koller et al., 2010a) is, we believe, an interesting avenue for future research.

Acknowledgments. We are grateful to Jörg Hoffmann for improving the efficiency of FF in the SCRISP domain at a crucial time, and to Margaret Mitchell, Matthew Stone and Kees van Deemter for helping us expand our view of the context-dependent adjective generation problem. We also thank Ines Rehbein and Josef Ruppenhofer for testing early implementations of our system, and Andrew Gargett as well as the reviewers for their helpful comments.

References

Douglas E. Appelt. 1985. *Planning English sentences*. Cambridge University Press, Cambridge, England.

Carlos Areces, Alexander Koller, and Kristina Striegnitz. 2008. Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference*, pages 42–49, Salt Fork, Ohio, USA.

Luciana Benotti. 2009. Clarification potential of instructions. In *Proceedings of the SIGDIAL 2009 Conference*, pages 196–205, London, UK.

Michael Brenner and Ivana Kruijff-Korbayová. 2008. A continual multiagent planning approach to situated dialogue. In *Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue*, London, UK.

David Chen and Igor Karpov. 2009. The GIVE-1 Austin system. In *The First GIVE Challenge: System descriptions*. `http://www.give-challenge.org/research/files/GIVE-09-Austin.pdf`.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19.

Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. 2009. Semantic attachments for domain-independent planning systems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*, pages 114–121.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Jörg Hoffmann. 2002. Extending FF to numerical state variables. In *Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, Berlin, Germany.

Hans Kamp and Barbara Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129 – 191.

Alexander Koller and Jörg Hoffmann. 2010. Waking up a sleeping rabbit: On natural-language sentence generation with FF. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, Toronto, Canada.

Alexander Koller and Matthew Stone. 2007. Sentence generation as planning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic.

Alexander Koller, Andrew Gargett, and Konstantina Garoufi. 2010a. A scalable model of planning perlocutionary acts. In *Proceedings of the 14th Workshop on the Semantics and Pragmatics of Dialogue*, Poznan, Poland.

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010b. The First Challenge on Generating Instructions in Virtual Environments. In M. Theune and E. Krahmer, editors, *Empirical Methods in Natural Language Generation*, volume 5790 of *LNCS*, pages 337–361. Springer, Berlin/Heidelberg. To appear.

Emiel Krahmer and Mariet Theune. 2002. Efficient context-sensitive generation of referring expressions. In Kees van Deemter and Rodger Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pages 223–264. CSLI Publications.

Emiel Krahmer, Sebastiaan van Erk, and André Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Margaret Mitchell. 2009. Class-based ordering of prenominal modifiers. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 50–57, Athens, Greece.

Dana Nau, Malik Ghallab, and Paolo Traverso. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

C. Raymond Perrault and James F. Allen. 1980. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3–4):167–182.

Paul Portner. 2007. Imperatives and modals. *Natural Language Semantics*, 15(4):351–383.

James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 135–143, College Park, Maryland, USA.

Mark Steedman and Ronald P. A. Petrick. 2007. Planning dialog actions. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 265–272, Antwerp, Belgium.

Laura Stoia, Donna K. Byron, Darla Magdalene Shockley, and Eric Fosler-Lussier. 2006. Sentence planning for realtime navigational instructions. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL*, pages 157–160, Morristown, NJ, USA.

Laura Stoia, Darla M. Shockley, Donna K. Byron, and Eric Fosler-Lussier. 2008. SCARE: A situated corpus with annotated referring expressions. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco.

Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. 2003. Microplanning with communicative intentions: The SPUD system. *Computational Intelligence*, 19(4):311–381.

Kees van Deemter. 2006. Generating referring expressions that involve gradable properties. *Computational Linguistics*, 32(2).

Beyond NomBank: A Study of Implicit Arguments for Nominal Predicates

Matthew Gerber and **Joyce Y. Chai**
Department of Computer Science
Michigan State University
East Lansing, Michigan, USA
{gerberm2,jchai}@cse.msu.edu

Abstract

Despite its substantial coverage, NomBank does not account for all within-sentence arguments and ignores extra-sentential arguments altogether. These arguments, which we call *implicit*, are important to semantic processing, and their recovery could potentially benefit many NLP applications. We present a study of implicit arguments for a select group of frequent nominal predicates. We show that implicit arguments are pervasive for these predicates, adding 65% to the coverage of NomBank. We demonstrate the feasibility of recovering implicit arguments with a supervised classification model. Our results and analyses provide a baseline for future work on this emerging task.

1 Introduction

Verbal and nominal semantic role labeling (SRL) have been studied independently of each other (Carreras and Màrquez, 2005; Gerber et al., 2009) as well as jointly (Surdeanu et al., 2008; Hajič et al., 2009). These studies have demonstrated the maturity of SRL within an evaluation setting that restricts the argument search space to the sentence containing the predicate of interest. However, as shown by the following example from the Penn TreeBank (Marcus et al., 1993), this restriction excludes extra-sentential arguments:

(1) [arg_0 The two companies] [$pred$ **produce**] [arg_1 market pulp, containerboard and white paper]. The goods could be manufactured closer to customers, saving [$pred$ **shipping**] costs.

The first sentence in Example 1 includes the PropBank (Kingsbury et al., 2002) analysis of the verbal predicate *produce*, where arg_0 is the agentive producer and arg_1 is the produced entity. The second sentence contains an instance of the nominal predicate *shipping* that is not associated with arguments in NomBank (Meyers, 2007).

From the sentences in Example 1, the reader can infer that *The two companies* refers to the agents (arg_0) of the *shipping* predicate. The reader can also infer that *market pulp, containerboard and white paper* refers to the shipped entities (arg_1 of *shipping*).[1] These extra-sentential arguments have not been annotated for the *shipping* predicate and cannot be identified by a system that restricts the argument search space to the sentence containing the predicate. NomBank also ignores many within-sentence arguments. This is shown in the second sentence of Example 1, where *The goods* can be interpreted as the arg_1 of *shipping*. These examples demonstrate the presence of arguments that are not included in NomBank and cannot easily be identified by systems trained on the resource. We refer to these arguments as *implicit*.

This paper presents our study of implicit arguments for nominal predicates. We began our study by annotating implicit arguments for a select group of predicates. For these predicates, we found that implicit arguments add 65% to the existing role coverage of NomBank.[2] This increase has implications for tasks (e.g., question answering, information extraction, and summarization) that benefit from semantic analysis. Using our annotations, we constructed a feature-based model for automatic implicit argument identification that unifies standard verbal and nominal SRL. Our results indicate a 59% relative (15-point absolute) gain in F_1 over an informed baseline. Our analyses highlight strengths and weaknesses of the approach, providing insights for future work on this emerging task.

[1] In PropBank and NomBank, the interpretation of each role (e.g., arg_0) is specific to a predicate sense.

[2] Role coverage indicates the percentage of roles filled.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1583–1592,
Uppsala, Sweden, 11-16 July 2010.

In the following section, we review related research, which is historically sparse but recently gaining traction. We present our annotation effort in Section 3, and follow with our implicit argument identification model in Section 4. In Section 5, we describe the evaluation setting and present our experimental results. We analyze these results in Section 6 and conclude in Section 7.

2 Related work

Palmer et al. (1986) made one of the earliest attempts to automatically recover extra-sentential arguments. Their approach used a fine-grained domain model to assess the compatibility of candidate arguments and the slots needing to be filled.

A phenomenon similar to the implicit argument has been studied in the context of Japanese anaphora resolution, where a missing case-marked constituent is viewed as a zero-anaphoric expression whose antecedent is treated as the implicit argument of the predicate of interest. This behavior has been annotated manually by Iida et al. (2007), and researchers have applied standard SRL techniques to this corpus, resulting in systems that are able to identify missing case-marked expressions in the surrounding discourse (Imamura et al., 2009). Sasano et al. (2004) conducted similar work with Japanese indirect anaphora. The authors used automatically derived nominal case frames to identify antecedents. However, as noted by Iida et al., grammatical cases do not stand in a one-to-one relationship with semantic roles in Japanese (the same is true for English).

Fillmore and Baker (2001) provided a detailed case study of implicit arguments (termed *null instantiations* in that work), but did not provide concrete methods to account for them automatically. Previously, we demonstrated the importance of filtering out nominal predicates that take no local arguments (Gerber et al., 2009); however, this work did not address the identification of implicit arguments. Burchardt et al. (2005) suggested approaches to implicit argument identification based on observed coreference patterns; however, the authors did not implement and evaluate such methods. We draw insights from all three of these studies. We show that the identification of implicit arguments for nominal predicates leads to fuller semantic interpretations when compared to traditional SRL methods. Furthermore, motivated by Burchardt et al., our model uses a quantitative analysis of naturally occurring coreference patterns to aid implicit argument identification.

Most recently, Ruppenhofer et al. (2009) conducted SemEval Task 10, "Linking Events and Their Participants in Discourse", which evaluated implicit argument identification systems over a common test set. The task organizers annotated implicit arguments across entire passages, resulting in data that cover many distinct predicates, each associated with a small number of annotated instances. In contrast, our study focused on a select group of nominal predicates, each associated with a large number of annotated instances.

3 Data annotation and analysis

3.1 Data annotation

Implicit arguments have not been annotated within the Penn TreeBank, which is the textual and syntactic basis for NomBank. Thus, to facilitate our study, we annotated implicit arguments for instances of nominal predicates within the standard training, development, and testing sections of the TreeBank. We limited our attention to nominal predicates with unambiguous role sets (i.e., senses) that are derived from verbal role sets. We then ranked this set of predicates using two pieces of information: (1) the average difference between the number of roles expressed in nominal form (in NomBank) versus verbal form (in PropBank) and (2) the frequency of the nominal form in the corpus. We assumed that the former gives an indication as to how many implicit roles an instance of the nominal predicate might have. The product of (1) and (2) thus indicates the potential prevalence of implicit arguments for a predicate. To focus our study, we ranked the predicates in NomBank according to this product and selected the top ten, shown in Table 1.

We annotated implicit arguments document-by-document, selecting all singular and plural nouns derived from the predicates in Table 1. For each missing argument position of each predicate instance, we inspected the local discourse for a suitable implicit argument. We limited our attention to the current sentence as well as all preceding sentences in the document, annotating all mentions of an implicit argument within this window.

In the remainder of this paper, we will use $iarg_n$ to refer to an implicit argument position n. We will use arg_n to refer to an argument provided by PropBank or NomBank. We will use p to mark

		Pre-annotation			Post-annotation	
			Role average			
Predicate	#	Role coverage (%)	Noun	Verb	Role coverage (%)	Noun role average
price	217	42.4	1.7	1.7	55.3	2.2
sale	185	24.3	1.2	2.0	42.0	2.1
investor	160	35.0	1.1	2.0	54.6	1.6
fund	109	8.7	0.4	2.0	21.6	0.9
loss	104	33.2	1.3	2.0	46.9	1.9
plan	102	30.9	1.2	1.8	49.3	2.0
investment	102	15.7	0.5	2.0	33.3	1.0
cost	101	26.2	1.1	2.3	47.5	1.9
bid	88	26.9	0.8	2.2	72.0	2.2
loan	85	22.4	1.1	2.5	41.2	2.1
Overall	1,253	28.0	1.1	2.0	46.2	1.8

Table 1: Predicates targeted for annotation. The second column gives the number of predicate instances annotated. *Pre-annotation* numbers only include NomBank annotations, whereas *Post-annotation* numbers include NomBank and implicit argument annotations. *Role coverage* indicates the percentage of roles filled. *Role average* indicates how many roles, on average, are filled for an instance of a predicate's noun form or verb form within the TreeBank. Verbal role averages were computed using PropBank.

predicate instances. Below, we give an example annotation for an instance of the *investment* predicate:

(2) [$iarg_0$ Participants] will be able to transfer [$iarg_1$ money] to [$iarg_2$ other investment funds]. The [p **investment**] choices are limited to [$iarg_2$ a stock fund and a money-market fund].

NomBank does not associate this instance of *investment* with any arguments; however, we were able to identify the investor ($iarg_0$), the thing invested ($iarg_1$), and two mentions of the thing invested in ($iarg_2$).

Our data set was also independently annotated by an undergraduate linguistics student. For each missing argument position, the student was asked to identify the closest acceptable implicit argument within the current and preceding sentences. The argument position was left unfilled if no acceptable constituent could be found. For a missing argument position, the student's annotation agreed with our own if both identified the same constituent or both left the position unfilled. Analysis indicated an agreement of 67% using Cohen's kappa coefficient (Cohen, 1960).

3.2 Annotation analysis

Role coverage for a predicate instance is equal to the number of filled roles divided by the number of roles in the predicate's lexicon entry. Role coverage for the marked predicate in Example 2 is 0/3 for NomBank-only arguments and 3/3 when the annotated implicit arguments are also considered. Returning to Table 1, the third column gives role coverage percentages for NomBank-only arguments. The sixth column gives role coverage percentages when both NomBank arguments and the annotated implicit arguments are considered. Overall, the addition of implicit arguments created a 65% relative (18-point absolute) gain in role coverage across the 1,253 predicate instances that we annotated.

The predicates in Table 1 are typically associated with fewer arguments on average than their corresponding verbal predicates. When considering NomBank-only arguments, this difference (compare columns four and five) varies from zero (for *price*) to a factor of five (for *fund*). When implicit arguments are included in the comparison, these differences are reduced and many nominal predicates express approximately the same number of arguments on average as their verbal counterparts (compare the fifth and seventh columns).

In addition to role coverage and average count, we examined the location of implicit arguments. Figure 1 shows that approximately 56% of the implicit arguments in our data can be resolved within the sentence containing the predicate. The remaining implicit arguments require up to forty-six sen-

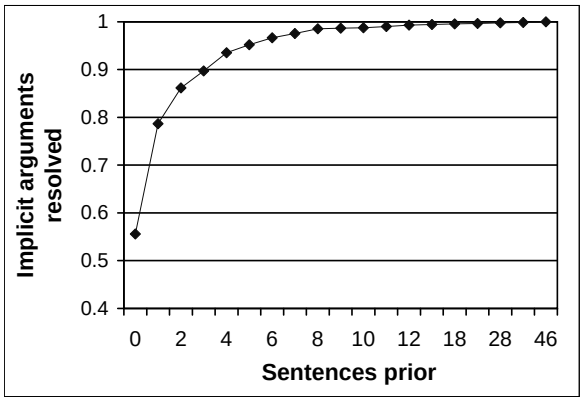


Figure 1: Location of implicit arguments. For missing argument positions with an implicit filler, the y-axis indicates the likelihood of the filler being found at least once in the previous x sentences.

tences for resolution; however, a vast majority of these can be resolved within the previous few sentences. Section 6 discusses implications of this skewed distribution.

4 Implicit argument identification

4.1 Model formulation

In our study, we assumed that each sentence in a document had been analyzed for PropBank and NomBank predicate-argument structure. NomBank includes a lexicon listing the possible argument positions for a predicate, allowing us to identify missing argument positions with a simple lookup. Given a nominal predicate instance p with a missing argument position $iarg_n$, the task is to search the surrounding discourse for a constituent c that fills $iarg_n$. Our model conducts this search over all constituents annotated by either PropBank or NomBank with non-adjunct labels.

A candidate constituent c will often form a coreference chain with other constituents in the discourse. Consider the following abridged sentences, which are adjacent in their Penn TreeBank document:

(3) [Mexico] desperately needs investment.

(4) Conservative Japanese investors are put off by [Mexico's] investment regulations.

(5) Japan is the fourth largest investor in [c Mexico], with 5% of the total [p **investments**].

NomBank does not associate the labeled instance of *investment* with any arguments, but it is clear from the surrounding discourse that constituent c (referring to Mexico) is the thing being invested in (the $iarg_2$). When determining whether c is the $iarg_2$ of *investment*, one can draw evidence from other mentions in c's coreference chain. Example 3 states that Mexico needs investment. Example 4 states that Mexico regulates investment. These propositions, which can be derived via traditional SRL analyses, should increase our confidence that c is the $iarg_2$ of *investment* in Example 5.

Thus, the unit of classification for a candidate constituent c is the three-tuple $\langle p, iarg_n, c' \rangle$, where c' is a coreference chain comprising c and its coreferent constituents.[3] We defined a binary classification function $Pr(+|\langle p, iarg_n, c' \rangle)$ that predicts the probability that the entity referred to by c fills the missing argument position $iarg_n$ of predicate instance p. In the remainder of this paper, we will refer to c as the *primary filler*, differentiating it from other mentions in the coreference chain c'. In the following section, we present the feature set used to represent each three-tuple within the classification function.

4.2 Model features

Starting with a wide range of features, we performed floating forward feature selection (Pudil et al., 1994) over held-out development data comprising implicit argument annotations from section 24 of the Penn TreeBank. As part of the feature selection process, we conducted a grid search for the best per-class cost within LibLinear's logistic regression solver (Fan et al., 2008). This was done to reduce the negative effects of data imbalance, which is severe even when selecting candidates from the current and previous few sentences. Table 2 shows the selected features, which are quite different from those used in our previous work to identify traditional semantic arguments (Gerber et al., 2009).[4] Below, we give further explanations for some of the features.

Feature 1 models the semantic role relationship between each mention in c' and the missing argument position $iarg_n$. To reduce data sparsity, this feature generalizes predicates and argument positions to their VerbNet (Kipper, 2005) classes and

[3] We used OpenNLP for coreference identification: http://opennlp.sourceforge.net

[4] We have omitted many of the lowest-ranked features. Descriptions of these features can be obtained by contacting the authors.

#	Feature value description
1*	For every f, the VerbNet class/role of p_f/arg_f concatenated with the class/role of p/$iarg_n$.
2*	Average pointwise mutual information between $\langle p, iarg_n \rangle$ and any $\langle p_f, arg_f \rangle$.
3	Percentage of all f that are definite noun phrases.
4	Minimum absolute sentence distance from any f to p.
5*	Minimum pointwise mutual information between $\langle p, iarg_n \rangle$ and any $\langle p_f, arg_f \rangle$.
6	Frequency of the nominal form of p within the document that contains it.
7	Nominal form of p concatenated with $iarg_n$.
8	Nominal form of p concatenated with the sorted integer argument indexes from all arg_n of p.
9	Number of mentions in c'.
10*	Head word of p's right sibling node.
11	For every f, the synset (Fellbaum, 1998) for the head of f concatenated with p and $iarg_n$.
12	Part of speech of the head of p's parent node.
13	Average absolute sentence distance from any f to p.
14*	Discourse relation whose two discourse units cover c (the primary filler) and p.
15	Number of left siblings of p.
16	Whether p is the head of its parent node.
17	Number of right siblings of p.

Table 2: Features for determining whether c fills $iarg_n$ of predicate p. For each mention f (denoting a $f$iller) in the coreference chain c', we define p_f and arg_f to be the predicate and argument position of f. Features are sorted in descending order of feature selection gain. Unless otherwise noted, all predicates were normalized to their verbal form and all argument positions (e.g., arg_n and $iarg_n$) were interpreted as labels instead of word content. Features marked with an asterisk are explained in Section 4.2.

semantic roles using SemLink.[5] For explanation purposes, consider again Example 1, where we are trying to fill the $iarg_0$ of *shipping*. Let c' contain a single mention, *The two companies*, which is the arg_0 of *produce*. As described in Table 2, feature 1 is instantiated with a value of *create.agent-send.agent*, where *create* and *send* are the VerbNet classes that contain *produce* and *ship*, respectively. In the conversion to LibLinear's instance representation, this instantiation is converted into a single binary feature *create.agent-send.agent* whose value is one. Features 1 and 11 are instantiated once for each mention in c', allowing the model to consider information from multiple mentions of the same entity.

Features 2 and 5 are inspired by the work of Chambers and Jurafsky (2008), who investigated unsupervised learning of narrative event sequences using pointwise mutual information (PMI) between syntactic positions. We used a similar PMI score, but defined it with respect to semantic arguments instead of syntactic dependencies. Thus, the values for features 2 and 5 are computed as follows (the notation is explained in the caption for Table 2):

$$pmi(\langle p, iarg_n \rangle, \langle p_f, arg_f \rangle) = log \frac{P_{coref}(\langle p, iarg_n \rangle, \langle p_f, arg_f \rangle)}{P_{coref}(\langle p, iarg_n \rangle, *) P_{coref}(\langle p_f, arg_f \rangle, *)} \quad (6)$$

To compute Equation 6, we first labeled a subset of the Gigaword corpus (Graff, 2003) using the verbal SRL system of Punyakanok et al. (2008) and the nominal SRL system of Gerber et al. (2009). We then identified coreferent pairs of arguments using OpenNLP. Suppose the resulting data has N coreferential pairs of argument positions. Also suppose that M of these pairs comprise $\langle p, arg_n \rangle$ and $\langle p_f, arg_f \rangle$. The numerator in Equation 6 is defined as $\frac{M}{N}$. Each term in the denominator is obtained similarly, except that M is computed as the total number of coreference pairs comprising an argument position (e.g., $\langle p, arg_n \rangle$) and *any* other argument position. Like Chambers and Jurafsky, we also used the discounting method suggested by Pantel and Ravichandran (2004) for low-frequency observations. The PMI score is somewhat noisy due to imperfect output, but it provides information that is useful for classification.

[5] http://verbs.colorado.edu/semlink

Feature 10 does not depend on c' and is specific to each predicate. Consider the following example:

(7) Statistics Canada reported that its [arg_1 industrial-product] [p **price**] index dropped 2% in September.

The "[p price] index" collocation is rarely associated with an arg_0 in NomBank or with an $iarg_0$ in our annotations (both argument positions denote the seller). Feature 10 accounts for this type of behavior by encoding the syntactic head of p's right sibling. The value of feature 10 for Example 7 is *price:index*. Contrast this with the following:

(8) [$iarg_0$ The company] is trying to prevent further [p **price**] drops.

The value of feature 10 for Example 8 is *price:drop*. This feature captures an important distinction between the two uses of *price*: the former rarely takes an $iarg_0$, whereas the latter often does. Features 12 and 15-17 account for predicate-specific behaviors in a similar manner.

Feature 14 identifies the discourse relation (if any) that holds between the candidate constituent c and the filled predicate p. Consider the following example:

(9) [$iarg_0$ SFE Technologies] reported a net loss of $889,000 on sales of $23.4 million.

(10) That compared with an operating [p **loss**] of [arg_1 $1.9 million] on sales of $27.4 million in the year-earlier period.

In this case, a *comparison* discourse relation (signaled by the underlined text) holds between the first and sentence sentence. The coherence provided by this relation encourages an inference that identifies the marked $iarg_0$ (the loser). Throughout our study, we used gold-standard discourse relations provided by the Penn Discourse TreeBank (Prasad et al., 2008).

5 Evaluation

We trained the feature-based logistic regression model over 816 annotated predicate instances associated with 650 implicitly filled argument positions (not all predicate instances had implicit arguments). During training, a candidate three-tuple $\langle p, iarg_n, c' \rangle$ was given a positive label if the candidate implicit argument c (the primary filler) was annotated as filling the missing argument position. To factor out errors from standard SRL analyses, the model used gold-standard argument labels provided by PropBank and NomBank. As shown in Figure 1 (Section 3.2), implicit arguments tend to be located in close proximity to the predicate. We found that using all candidate constituents c within the current and previous two sentences worked best on our development data.

We compared our supervised model with the simple baseline heuristic defined below:[6]

> Fill $iarg_n$ for predicate instance p with the nearest constituent in the two-sentence candidate window that fills arg_n for a different instance of p, where all nominal predicates are normalized to their verbal forms.

The normalization allows an existing arg_0 for the verb *invested* to fill an $iarg_0$ for the noun *investment*. We also evaluated an oracle model that made gold-standard predictions for candidates within the two-sentence prediction window.

We evaluated these models using the methodology proposed by Ruppenhofer et al. (2009). For each missing argument position of a predicate instance, the models were required to either (1) identify a single constituent that fills the missing argument position or (2) make no prediction and leave the missing argument position unfilled. We scored predictions using the Dice coefficient, which is defined as follows:

$$\frac{2 * |Predicted \bigcap True|}{|Predicted| + |True|} \quad (11)$$

$Predicted$ is the set of tokens subsumed by the constituent predicted by the model as filling a missing argument position. $True$ is the set of tokens from a single annotated constituent that fills the missing argument position. The model's prediction receives a score equal to the maximum Dice overlap across any one of the annotated fillers. Precision is equal to the summed prediction scores divided by the number of argument positions filled by the model. Recall is equal to the summed prediction scores divided by the number of argument positions filled in our annotated data. Predictions not covering the head of a true filler were assigned a score of zero.

[6] This heuristic outperformed a more complicated heuristic that relied on the PMI score described in section 4.2.

			Baseline			Discriminative				Oracle	
	#	Imp. #	P	R	F_1	P	R	F_1	p	R	F_1
sale	64	60	50.0	28.3	36.2	47.2	41.7	**44.2**	0.118	80.0	88.9
price	121	53	24.0	11.3	15.4	36.0	32.6	**34.2**	0.008	88.7	94.0
investor	78	35	33.3	5.7	9.8	36.8	40.0	**38.4**	< 0.001	91.4	95.5
bid	19	26	100.0	19.2	**32.3**	23.8	19.2	21.3	0.280	57.7	73.2
plan	25	20	83.3	25.0	38.5	78.6	55.0	**64.7**	0.060	82.7	89.4
cost	25	17	66.7	23.5	34.8	61.1	64.7	**62.9**	0.024	94.1	97.0
loss	30	12	71.4	41.7	52.6	83.3	83.3	**83.3**	0.020	100.0	100.0
loan	11	9	50.0	11.1	18.2	42.9	33.3	**37.5**	0.277	88.9	94.1
investment	21	8	0.0	0.0	0.0	40.0	25.0	**30.8**	0.182	87.5	93.3
fund	43	6	0.0	0.0	0.0	14.3	16.7	**15.4**	0.576	50.0	66.7
Overall	437	246	48.4	18.3	26.5	44.5	40.4	**42.3**	< 0.001	83.1	90.7

Table 3: Evaluation results. The second column gives the number of predicate instances evaluated. The third column gives the number of ground-truth implicitly filled argument positions for the predicate instances (not all instances had implicit arguments). P, R, and F_1 indicate precision, recall, and F-measure ($\beta = 1$), respectively. p-values denote the bootstrapped significance of the difference in F_1 between the baseline and discriminative models. Oracle precision (not shown) is 100% for all predicates.

Our evaluation data comprised 437 predicate instances associated with 246 implicitly filled argument positions. Table 3 presents the results. Predicates with the highest number of implicit arguments - *sale* and *price* - showed F_1 increases of 8 points and 18.8 points, respectively. Overall, the discriminative model increased F_1 performance 15.8 points (59.6%) over the baseline.

We measured human performance on this task by running our undergraduate assistant's annotations against the evaluation data. Our assistant achieved an overall F_1 score of 58.4% using the same candidate window as the baseline and discriminative models. The difference in F_1 between the discriminative and human results had an exact p-value of less than 0.001. All significance testing was performed using a two-tailed bootstrap method similar to the one described by Efron and Tibshirani (1993).

6 Discussion

6.1 Feature ablation

We conducted an ablation study to measure the contribution of specific feature sets. Table 4 presents the ablation configurations and results. For each configuration, we retrained and retested the discriminative model using the features described. As shown, we observed significant losses when excluding features that relate the semantic roles of mentions in c' to the semantic role of the missing argument position (first configuration). The second configuration tested the effect of using *only* the SRL-based features. This also resulted in significant performance losses, suggesting that the other features contribute useful information. Lastly, we tested the effect of removing discourse relations (feature 14), which are likely to be difficult to extract reliably in a practical setting. As shown, this feature did not have a statistically significant effect on performance and could be excluded in future applications of the model.

	Percent change (p-value)		
Configuration	P	R	F_1
Remove 1,2,5	-35.3 (< 0.01)	-36.1 (< 0.01)	-35.7 (< 0.01)
Use 1,2,5 only	-26.3 (< 0.01)	-11.9 (0.05)	-19.2 (< 0.01)
Remove 14	0.2 (0.95)	1.0 (0.66)	0.7 (0.73)

Table 4: Feature ablation results. The first column lists the feature configurations. All changes are percentages relative to the full-featured discriminative model. p-values for the changes are indicated in parentheses.

6.2 Unclassified true implicit arguments

Of all the errors made by the system, approximately 19% were caused by the system's failure to

generate a candidate constituent c that was a correct implicit argument. Without such a candidate, the system stood no chance of identifying a correct implicit argument. Two factors contributed to this type of error, the first being our assumption that implicit arguments are also core (i.e., arg_n) arguments to traditional SRL structures. Approximately 8% of the overall error was due to a failure of this assumption. In many cases, the true implicit argument filled a non-core (i.e., adjunct) role within PropBank or NomBank.

More frequently, however, true implicit arguments were missed because the candidate window was too narrow. This accounts for 12% of the overall error. Oracle recall (second-to-last column in Table 3) indicates the nominals that suffered most from windowing errors. For example, the *sale* predicate was associated with the highest number of true implicit arguments, but only 80% of those could be resolved within the two-sentence candidate window. Empirically, we found that extending the candidate window uniformly for all predicates did not increase performance on the development data. The oracle results suggest that predicate-specific window settings might offer some advantage.

6.3 The *investment* and *fund* predicates

In Section 4.2, we discussed the *price* predicate, which frequently occurs in the "[p price] index" collocation. We observed that this collocation is rarely associated with either an overt arg_0 or an implicit $iarg_0$. Similar observations can be made for the *investment* and *fund* predicates. Although these two predicates are frequent, they are rarely associated with implicit arguments: *investment* takes only eight implicit arguments across its 21 instances, and *fund* takes only six implicit arguments across its 43 instances. This behavior is due in large part to collocations such as "[p investment] banker", "stock [p fund]", and "mutual [p fund]", which use predicate senses that are not eventive. Such collocations also violate our assumption that differences between the PropBank and NomBank argument structure for a predicate are indicative of implicit arguments (see Section 3.1 for this assumption).

Despite their lack of implicit arguments, it is important to account for predicates such as *investment* and *fund* because incorrect prediction of implicit arguments for them can lower precision. This is precisely what happened for the *fund* predicate, where the model incorrectly identified many implicit arguments for "stock [p fund]" and "mutual [p fund]". The left context of *fund* should help the model avoid this type of error; however, our feature selection process did not identify any overall gains from including this information.

6.4 Improvements versus the baseline

The baseline heuristic covers the simple case where identical predicates share arguments in the same position. Thus, it is interesting to examine cases where the baseline heuristic failed but the discriminative model succeeded. Consider the following sentence:

(12) Mr. Rogers recommends that [p **investors**] sell [$iarg_2$ takeover-related stock].

Neither NomBank nor the baseline heuristic associate the marked predicate in Example 12 with any arguments; however, the feature-based model was able to correctly identify the marked $iarg_2$ as the entity being invested in. This inference captured a tendency of investors to sell the things they have invested in.

We conclude our discussion with an example of an extra-sentential implicit argument:

(13) [$iarg_0$ Olivetti] has denied that it violated the rules, asserting that the shipments were properly licensed. However, the legality of these [p **sales**] is still an open question.

As shown in Example 13, the system was able to correctly identify *Olivetti* as the agent in the selling event of the second sentence. This inference involved two key steps. First, the system identified coreferent mentions of *Olivetti* that participated in exporting and supplying events (not shown). Second, the system identified a tendency for exporters and suppliers to also be sellers. Using this knowledge, the system extracted information that could not be extracted by the baseline heuristic or a traditional SRL system.

7 Conclusions and future work

Current SRL approaches limit the search for arguments to the sentence containing the predicate of interest. Many systems take this assumption a step further and restrict the search to the predicate's local syntactic environment; however, predicates and the sentences that contain them rarely

exist in isolation. As shown throughout this paper, they are usually embedded in a coherent and semantically rich discourse that must be taken into account. We have presented a preliminary study of implicit arguments for nominal predicates that focused specifically on this problem.

Our contribution is three-fold. First, we have created gold-standard implicit argument annotations for a small set of pervasive nominal predicates.[7] Our analysis shows that these annotations add 65% to the role coverage of NomBank. Second, we have demonstrated the feasibility of recovering implicit arguments for many of the predicates, thus establishing a baseline for future work on this emerging task. Third, our study suggests a few ways in which this research can be moved forward. As shown in Section 6, many errors were caused by the absence of true implicit arguments within the set of candidate constituents. More intelligent windowing strategies in addition to alternate candidate sources might offer some improvement. Although we consistently observed development gains from using automatic coreference resolution, this process creates errors that need to be studied more closely. It will also be important to study implicit argument patterns of non-verbal predicates such as the partitive *percent*. These predicates are among the most frequent in the TreeBank and are likely to require approaches that differ from the ones we pursued.

Finally, any extension of this work is likely to encounter a significant knowledge acquisition bottleneck. Implicit argument annotation is difficult because it requires both argument and coreference identification (the data produced by Ruppenhofer et al. (2009) is similar). Thus, it might be productive to focus future work on (1) the extraction of relevant knowledge from existing resources (e.g., our use of coreference patterns from Gigaword) or (2) semi-supervised learning of implicit argument models from a combination of labeled and unlabeled data.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful questions and comments. We would also like to thank Malcolm Doering for his annotation effort. This work was supported in part by NSF grants IIS-0347548 and IIS-0840538.

[7] Our annotation data can be freely downloaded at http://links.cse.msu.edu:8000/lair/projects/semanticrole.html

References

Aljoscha Burchardt, Anette Frank, and Manfred Pinkal. 2005. Building text meaning representations from contextually related frames - a case study. In *Proceedings of the Sixth International Workshop on Computational Semantics*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics*, pages 789–797, Columbus, Ohio, June. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):3746.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May.

C.J. Fillmore and C.F. Baker. 2001. Frame semantics for text understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop, NAACL*.

Matthew Gerber, Joyce Y. Chai, and Adam Meyers. 2009. The role of implicit argumentation in nominal SRL. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 146–154, Boulder, Colorado, USA, June.

David Graff. 2003. English Gigaword. Linguistic Data Consortium, Philadelphia.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop in ACL-2007*, page 132139.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88, Suntec, Singapore, August. Association for Computational Linguistics.

P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT'02)*.

Karin Kipper. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Department of Computer and Information Science University of Pennsylvania.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn TreeBank. *Computational Linguistics*, 19:313–330.

Adam Meyers. 2007. Annotation guidelines for NomBank - noun argument structure for PropBank. Technical report, New York University.

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the 24th annual meeting on Association for Computational Linguistics*, pages 10–19, Morristown, NJ, USA. Association for Computational Linguistics.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Rashmi Prasad, Alan Lee, Nikhil Dinesh, Eleni Miltsakaki, Geraud Campion, Aravind Joshi, and Bonnie Webber. 2008. Penn discourse treebank version 2.0. Linguistic Data Consortium, February.

P. Pudil, J. Novovicova, and J. Kittler. 1994. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Comput. Linguist.*, 34(2):257–287.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado, June. Association for Computational Linguistics.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2004. Automatic construction of nominal case frames and its application to indirect anaphora resolution. In *Proceedings of Coling 2004*, pages 1201–1207, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Author Index